8 E-Commerce Platforms and Distributed Transaction Management

NOTE: This chapter covers specialized technologies for ecommerce and transaction processing. The reader may choose to postpone this discussion till later.

8.1 Introduction

Many enterprise architecture and integration projects involve ecommerce and transaction processing applications. Simply stated, ecommerce (EC) represents buying and selling over the Internet. At the core of ecommerce is on-line buying/selling through a catalog using a shopping cart, electronic wallet, or similar tool. Ecommerce includes both consumers purchasing goods and on-line buyers purchasing goods from a single supplier. It can also include links to back-end systems for inventory updates and credit checking. Technically speaking, ecommerce translates into the need for consumers to purchase items (C2B) and frequently to carry out business transactions across multiple organizations (B2B). In this chapter, we are primarily concerned with the details of the EC middleware needed to support EC transactions over the public Internet. At a very basic level, the following core EC functionalities need to be supported:

- Advertising
- Items Browsing, selection, items purchase cart management
- Purchasing
- Billing /invoicing
- Payments
- Shipping methods and inventory management for physical good

The infrastructure supporting these functionalities is subject to stringent security, performance and reliability requirements. A wide range of IT infrastructure components are needed to support these and other EC activities. Figure 8-1 shows a high level view of the IT infrastructure services needed to support variants of EC. This chapter concentrates on ***Commerce Servers*** that package several technologies (network support, EC middleware services, EC software development environments, and EC monitoring/control systems) to build, deploy and manage EC applications. These servers are also known as eCommerce servers (see the sidebar "Commerce Servers Versus Application Servers"). At a conceptual level, the IT infrastructure of the commerce servers consists of the following:

- **Networking services** to provide the network transport between EC partners. These services include the traditional routing and flow/error control support. The network services have been provided by private value added networks (VANs) but are now being provided through Public Internet and/or Extranets that use the Internet technologies over privately owned and/or managed networks. See Section 8.3.1 for more details.
- **General purpose Middleware services** to support interactions between remotely located, including but not restricted to, EC partners. These core components provide Web services, directory services (e.g., locating the wide range of EC services), electronic messaging (e.g., Email), remote data services (e.g., bulk data transfer, browsing, and programmed access), and remote application services (e.g., interactions between EC users and EC applications through remote procedure calls (RPCs), message oriented middleware (MOM), or Web Services calls). These services are not discussed in this chapter (they have been discussed in previous chapters).

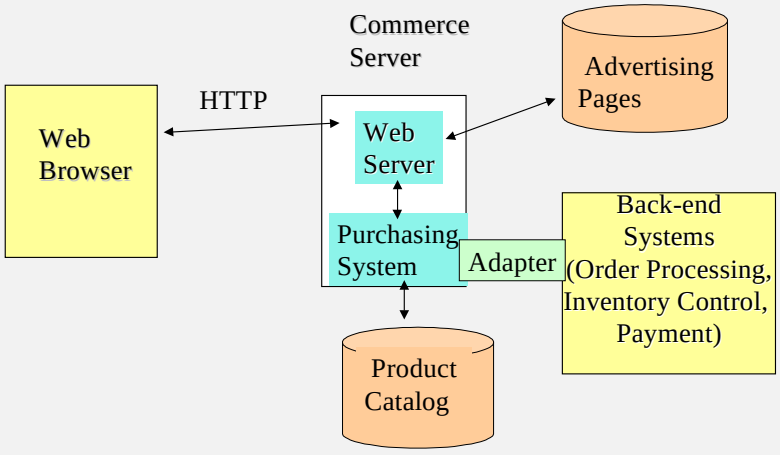
- **Ecommerce middleware services** to provide value added features needed by EC applications. Examples of these services are purchasing, payments, billing, EDI, and XML support. This middleware is the focus of this chapter. Additional details about most of these services can be found in Section 8.3.
- **B2B middleware services** needed to support the B2B trade. Examples of these services include: support for supply chain management, enterprise integration software, and electronic markets/trading hubs. These services are discussed in more detail in a later chapter.
- **Security issues in EC/EB** to assure that the information transfer between partners is conducted in a secure manner. We will discuss this topic briefly in Section 8.4**.**

Commerce Servers Versus Application Servers

The term ***commerce server*** (CS) is often used loosely in the industry to describe the set of subsystems bridging the gap between a web server and a payment server. Common functionality of a CS includes shopping carts, catalog management, purchasing logic (e.g., customer verification), logs/audit trails to track sales activities, and interfacing with the back-end enterprise systems such as payment, order processing, and inventory control. Sometimes the term is used to include the web server and/or the payment server itself. The following figure shows a conceptual view of commerce server.

Simply stated, an ***application server*** (also known as app server) is a platform for development, deployment, and management/support of Web-based applications. The current and future versions of application servers include facilities for development, deployment, and management of web-XML applications. This includes facilities for EJB (Enterprise Java Bean) component development, XML exchanges, load balancing, failure handling, and adapters for connecting to back-end applications.

In essence, a commerce server is an application server that specializes in e-commerce. An interesting illustrative example is the Netscape Application Server that combines web development capabilities with enterprise applications that integrate with corporate data sources. At present, this server has evolved into Sun Iplanet that provides a complete set of e-commerce facilities.



Conceptual View of a Commerce Sever

Many commerce servers are becoming commercially available. Microsoft's Commerce Server, Sun's Iplanet, and IBM's WebSphere are examples. It is not our objective to analyze commercial products in detail. Instead the building blocks of commerce servers are discussed in some detail.

Ecommerce systems rely very heavily on transaction processing systems to make sure that all data is properly synchronized and no transactions are lost due to system failures. Attachment A (Section 8.10) gives some technical details of transaction management, especially in distributed environments.

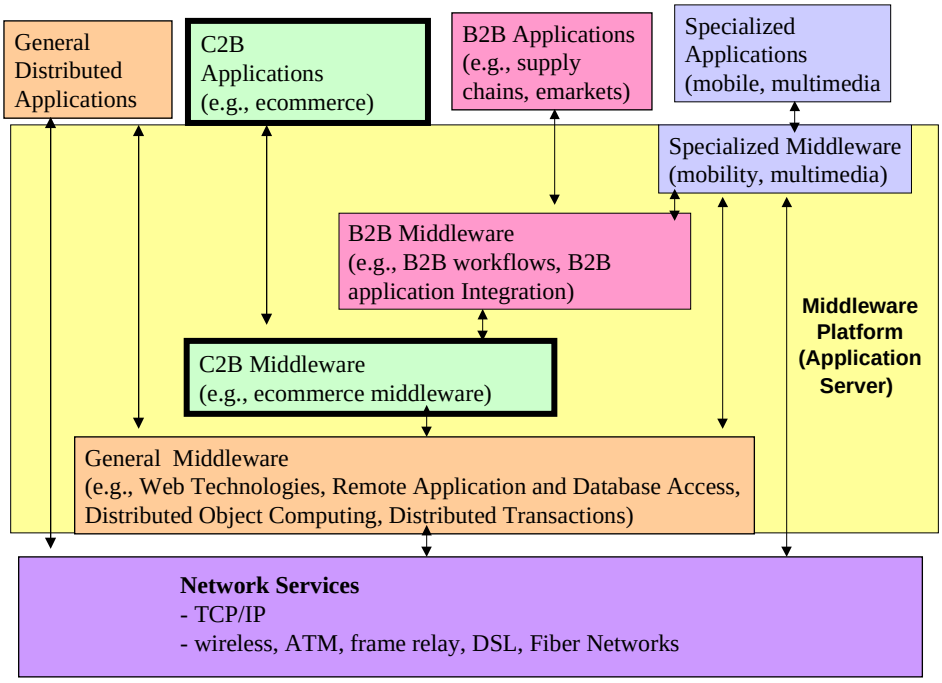
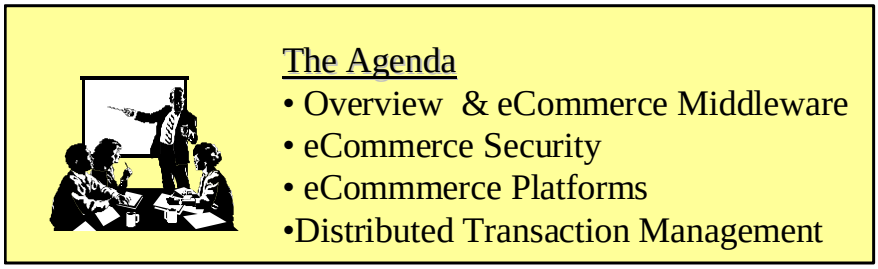


Figure 8-1: IT Infrastructure for E Commerce.



8.2 Internet-based Purchasing Overview

8.2.1 Case Study: Online Purchasing for XYZ Corp

XYZCorp wants to setup an on-line purchasing system that will allow customers to purchase the company products through the Web. You have been asked to workout the details to make it happen. Specifically, you have to develop an overall architecture of the system, show how the payment system will work through a credit card, identify the key players and their role in purchase order processing, develop a security solution, identify the role of XML in this system, and find an e-commerce platform that can support this system. We will provide hints about this at the end of the chapter.

- Get the demo copy of the selected e-commerce platform (availability of a demo copy may be a selection criteria), download it and run some simple experiments to see how these platforms work.
- What will you need to do to convert this storefront into a virtual storefront (i.e., the customer can choose items from multiple suppliers)?

8.2.2 A Quick Overview

Purchasing is at the core of Ecommerce. As shown in Figure 8-2, the purchasing process consists of several steps that can be viewed in terms of pre-purchase, purchase consummation, and post-purchase activities. EC middleware must support these activities over the Internet. Let us explain the principles of Internet-based purchasing through two simple examples. We will postpone, until later in this chapter, the discussion of more sophisticated purchasing models such as trading hubs, clearing houses, and electronic marketplaces.

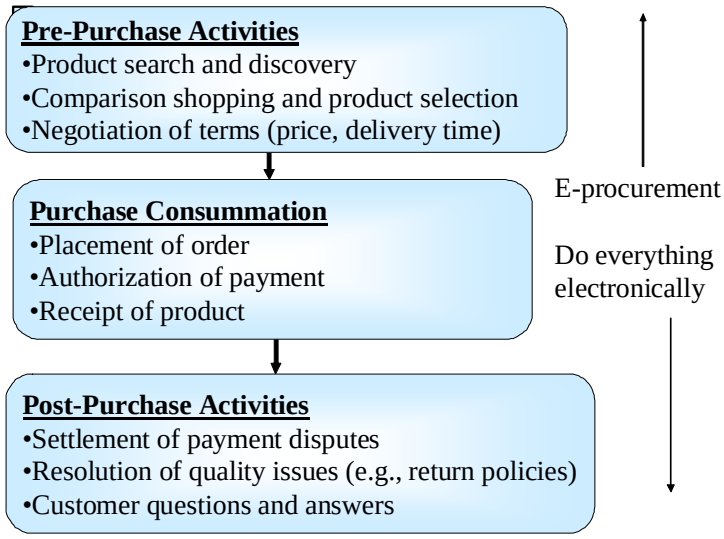


Figure 8-2: Purchasing Steps

8.2.3 A Simple C2B Purchasing Example

Let us start with a simple example of a company that wants to establish an electronic store front, i.e., allow customers to buy the products from the company over the Internet. The following discussion shows the usage scenarios and elaborates the principal activities of Ipurchase, a system developed to support the storefront. Figure 8-3 shows a simplified view of Ipurchase with various Web technologies such as HTTP, HTML, etc. The usage scenarios are presented from three different perspectives: customer usage, business administrator usage (e.g., purchasing department), and product administrator usage (e.g., IT Group).

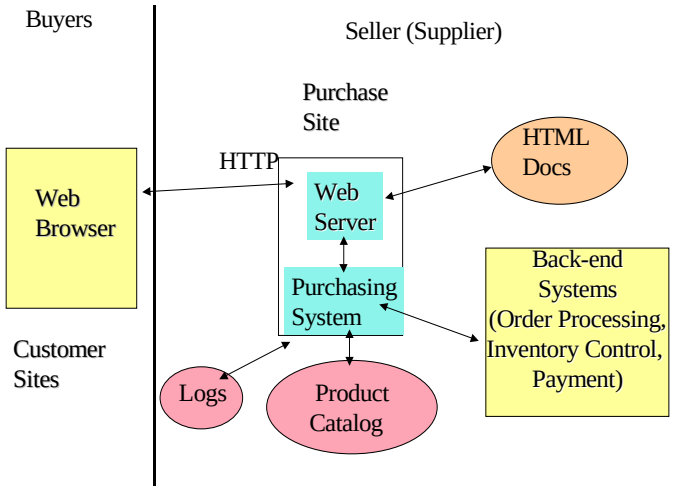


Figure 8-3: A Simple Internet-based Purchasing System

8.2.3.1 Customer Processing

i). Initial Processing

- Customer gets on the corporate Home Page.
- IPurchase Greeting screen shows and walks the customer through various informational and marketing pages.

ii). Search and Browse Catalog and Select Items

- A product review and selection screen is shown to the user as a default. Other views are also possible (e.g., vendor view, manufacturer view).
- Customer browses and/or searches through the catalog based on product attributes (e.g., price, name, manufacturer, etc.), synonyms, and full text searches (e.g., "find me a laptop").
- If an item is not available (as indicated in the catalog), the customer can choose to terminate the session or browse for other items.
- Customer selects the product(s) to purchase.
- A "shopping cart" is populated with the items selected by the customer.
- Customer verifies the items to be purchased and clicks on "purchase".
- The system asks for customer ID or payment information such as credit card number.
- The validation process is triggered to verify the payment information.
- Customer is notified whether to proceed (Catalog Review and Selection) or not with appropriate errors/guidance messages on how to correct the errors.
- The purchase information is also validated for exceeding the purchase limit. The buyer is given help in making corrections (shuffling the shopping cart), and resubmitting.

iii). Order Generation

- An order entry is created with a control number.
- The order log has information that can also be used for General Ledger (i.e., customer ID, name, items ordered, total quantity, etc.) .
- Order is logged in the log database.
- The order is sent to the order processing system.
- The user is given an end of order screen and is given the control number.
- An email may be also sent to the user documenting the order.

iv). Order Processing

- The order processing system receives the order and initiates its order processing.
- The supplier sends the needed items to the customer through its shipping department.
- If for some reason, the order cannot be fulfillled, then the system notifies the buyer through email or phone call.

8.2.3.2 Business Administrator Usage Scenarios

The business administrator (e.g., purchasing department) will be involved in the following steps to reconcile the orders and to print various reports. These operations will require password protection.

v). Reconciliation and customer support
- The order log database is reconciled with the shipment data.
- Error reports are generated for mismatches.
- Purchasing and shipment resolve the errors.
- Customer issues are resolved by the customer support department.

vi). Payment processing
- The payment is handled mainly through credit cards.
- High priced items are handled through purchase orders, invoices and accounts payable (i.e., accounts payable processes the invoices, and invokes other back-end processing such as General ledger).

viii). Management reporting
- Generate and send reports to managers periodically (who ordered what in their department).
- Allow ad hoc report generation on an ad hoc basis by the managers.
- Generate reports on system activities (how many people logged on, how many actually purchased something, etc.)

8.2.3.3 Product Administrator ("Web Master") Usage Scenarios

The administrators perform the following functions (after the system has been developed and installed):
- Purchase system installation and configuration procedures
- Design and populate the catalog
- Replenishment of the catalog periodically (e.g., daily)
- System performance monitoring, tuning, backup/recovery, etc.
- Update software when needed
- Audit the system for security breeches

8.2.4 A Simple B2B Purchasing Example

For sake of discussion, let us now introduce a simple B2B purchasing system, Bpurchase, that differs from the C2B Ipurchase system in two respects: a) it involves a B2B relationship between a buyer corporation and multiple suppliers, and b) the buyers are employees of the buying corporation with proper authorization for purchasing.

Bpurchase was developed for ordering low-cost, fast delivery (within 24 hours) items with a maximum order of $1,000 from multiple suppliers. We assume that less than a dozen suppliers (vendors) participate in the system handling around 40,000 orders per year. Bpurchase replaces a

current corporate purchase system for employees that is primarily phone-based. In the phone-based system, the employee calls the vendor, the vendor asks the employee few questions (e.g., items needed, employee ID, project number, etc.), supplies a control number to the employees, ships the requested items to the employee, and then sends an invoice to the company.

The purpose of Bpurchase is to minimize operational costs and improve customer satisfaction by automating the phone-based purchase system. The basic idea is to allow the employees to search and select the items to be purchased and create an electronic "purchase cart". From the purchase cart, an automatic order is generated that is sent to the vendors for purchasing.

The main B2B consideration is that an “open purchase order (PO)” agreement exists between the buying organization and the sellers, i.e., the sellers (vendors) send a monthly invoice to the buyer that shows all the purchases made that month. Open PO is a convenient way of buying low cost items such as office supplies (a PO does not have to be issued for each pencil).

The following discussion shows the same usage scenarios as discussed for the Ipurchase system. The usage scenarios are presented, as before, from three different perspectives: employee usage, business administrator usage (e.g., purchasing department), and product administrator usage (e.g., IT Group).

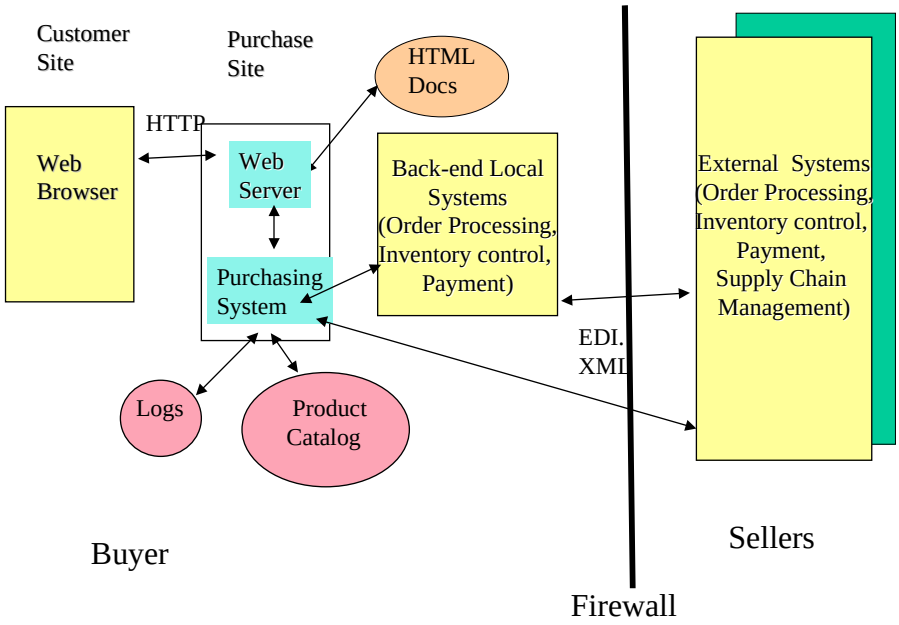


Figure 8-4: Conceptual View of B2B Purchasing

8.2.4.1 Employee Usage Scenarios

The employees will use the following steps to order supplies:

i). Initial Processing

- Employee gets on the corporate Home Page.
- BPurchase Greeting screen shows and asks for employee ID, project ID, etc. Passwords are not required to use the system.

- The validation process is triggered to verify employee and project information.
- Employee is notified whether to proceed (Catalog Review and Selection) or not with appropriate errors/guidance messages on how to correct the login errors.

ii). Search and Browse Catalog and Select Items

- A product review and selection screen is shown to the user as a default. Other views are also possible (e.g., vendor view, manufacturer view).
- Employee browses and/or searches through the catalog based on product attributes (e.g., price, name, manufacturer, etc.), synonyms, and full text searches (e.g., "find me adhesive tapes).
- If an item is not available (as indicated in the catalog), then the employee can optionally generate an email to the vendor asking about item availability. There is no back-order processing in this system. The user can search for the needed item from another vendor catalog or choose to terminate the session (i.e., call vendor off-line).
- Employee selects the product(s) to purchase.
- A "shopping cart" is populated with the items selected by the employee.
- Employee verifies the items to be purchased and clicks on "purchase".
- The purchase information is validated against the IPurchase limit, project information, budget, etc. The employee is given help in making corrections (shuffling the shopping cart), and resubmitting.

iii). Order Generation

- An order entry is created with a control number.
- The order log has information that can also be used for General Ledger (i.e., employee ID, name, project, task, items ordered, total quantity, vendor ordered from, etc.).
- Order is logged in the log database.
- The order is sent to the vendor through Email (other options for interfacing IPurchase with vendor systems are being investigated).
- The user is given an end of order screen and is given the control number.
- An email is sent to the user documenting the order.

iv). Vendor Processing

- The vendor receives the Email and initiates its order processing.
- The vendor sends the needed items to the employee.
- If for some reason, the vendor cannot fill the order, then the vendor notifies the employee through email or phone call.
- The invoices with additional details (e.g., what was purchased, who purchased it, etc.) are sent by the vendor to purchasing once a month. At present this information is sent on a floppy diskette.

8.2.4.2 Business Administrator Usage Scenarios

The business administrator (e.g., purchasing department) will be involved in the following steps to reconcile the orders and to print various reports. These operations will require password protection.

v). Reconciliation

- The order log database is reconciled with the vendor provided invoices.
- Error reports are generated for mismatches.
- Purchasing and vendors resolve the errors.

vi). Invoice processing

- The approved invoices are sent to accounts payable.
- Accounts payable processes the invoices.

- Other back-end processing (e.g., GL).

vii). Payments
- Accounts payable sends the checks to the vendors (once a month).

viii). Management reporting
- Generate and send reports to managers periodically (who ordered what in their department).
- Send an email to employees indicating what they have ordered and how much the company was billed for it (this will be used to verify that the employee has actually received the items).
- Allow ad hoc report generation on an ad hoc basis by the managers.
- Generate reports on system activities (how many people used it, how many through Web, etc.).

The business administrators also initialize and maintain the IPurchase system (e.g., define vendor information such as email and contact, define and modify validation rules (e.g., purchasing limits based on corporate rank), add/delete vendors, restrict vendors to certain products, etc.).

8.2.4.3 Product Administrator Usage Scenarios

The product administrators perform the following:
- System installation and configuration procedures
- Population of the consolidated catalog
- Replenishment of the consolidated catalog periodically (e.g., daily)
- System performance monitoring, tuning, backup/recovery, etc.

Finally, notice that, because users are authenticated against existing databases, we can eliminate the need to build and maintain a new system just to authenticate remote access users. This existing database can be Unix, Linux,, or Windows based.

Suggested Review Questions Before Proceeding

- Do you agree that e-commerce = online purchasing? Why and why not?
- What are the main steps in online purchasing and which ones have been most strongly influenced by the Internet?
- Suppose you want to sell office supplies through the Internet. Show ALL the steps you will go through to accomplish this.
- Develop 3 different views of online purchasing for small, medium, and large scale online purchasing. What are the common building blocks?
- Take a real-life online purchasing system that you are familiar with and compare/contrast it with the conceptual model described above.

8.3 e-Commerce Middleware

The EC middleware, as stated previously, provides value added features needed by the C2B applications and is at the heart of commerce servers. Examples of these services are shopping carts, catalog systems, electronic payment systems, customer care and billing, EDI/XML, and a variety of other services (see Figure 8-5). The EC middleware is briefly discussed below after a quick review of networks for EC.

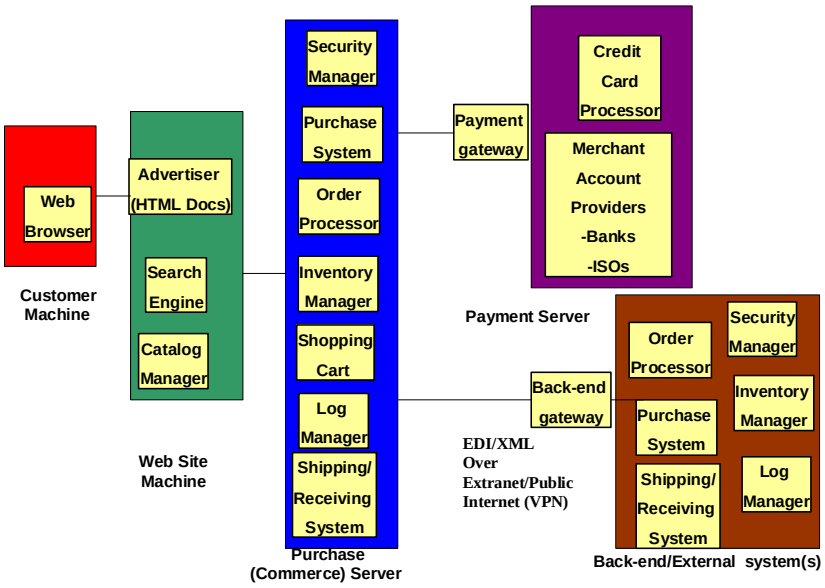


Figure 8-5: More Detailed View of e-Commerce

8.3.1 Extranets and Virtual Private Networks (VPNs) for ECommerce

"Extranet" or "enterprise intranets" are semi-private IP networks which are used to communicate within a group of interdependent communities of enterprises or trading partners. Examples of such a group of interdependent community would be the automotive industry (including parts suppliers, manufacturers, retailers, and insurers), the health care industry (including physicians, pharmacists, hospitals, labs, and health insurers), or the real estate industry (including brokers, lending agencies, insurers, lawyers, and inspectors). To succeed, Extranets need to support high quality EC services (e.g., advertising, browsing/selection, purchasing, billing, and payments) coupled with security and management considerations.

An *Extranet* consists of a collection of Internet segments (intranets), each protected by firewalls, which are interconnected using secure leased lines across the remote locations. This solution provides security and guaranteed bandwidth, at the cost of leasing lines from telecomm providers. In contrast, *Virtual Private Networks* (VPNs) achieve a similar goal (that is, securely connecting remote locations, branch offices, field workers, telecommuters, vendors, customers, and suppliers) using the public Internet instead of leased lines. This approach has the advantage of lower costs, and also allows occasional remote workers to reach a private Intranet from anywhere on the Internet. The main drawback is the Internet-quality effective bandwidth for the connection. The available bandwidth is further decreased because of the additional encryption step required for each IP packet.

Specifically, VPNs use IP tunnelling to provide security across protected intranet islands at the packet transport level. Tunnelling consists of a client-server pair that exchange encrypted packets using a private key derived from a token usually supplied by the user. The server sits at the border of a secure intranet and acts as a proxy towards clients that need to access the intranet from the outside. It provides authentication by recognizing user tokens and generating private keys for packet encryption. Packets are first compressed, then encrypted, and finally encapsulated into regular IP packets that are sent over the Internet. At the receiving end, the encapsulated packets are reconstructed and their integrity is verified (this is necessary because outsiders supposedly cannot understand the packets in transit, but they can still tamper with them). If the receiving end is a proxy, the packets are further routed to their destination inside the Intranet. The tunnel remains active for the duration of a session, and it is initially set-up when authentication takes place and private keys are exchanged. Frequently, schemes for one-time tokens with a user PIN are used.

In a typical usage scenario, a remote user connects to the Internet from anywhere, for example through a local ISP. Once on the network, the VPN client connects to the server at the Intranet location, and a tunnel is set-up between the client and the Intranet. The packets that travel through the tunnel are encrypted on one side and decrypted on the other side. Thus the tunnel provides a private and protected path between the client and the destination site.

8.3.2 Shopping Carts

The Shopping Cart functionality goes beyond simple rendering in the customer browser It provides the link between the purchasing action and the order merchant fulfillment. The website (e-commerce.about.com) has very useful information in this area.

The Web Store Software Selector at About.com offers information on over 40 shopping cart services categorized by price and listed with key details, such as whether the service is suitable for beginners and whether the shopping cart runs on the local server or the remote host's.

The “Real Soft Cart” is one example of such a Shopping Cart. The Real Soft Shopping Cart is an Internet ready to use shopping cart system. Real Soft integrates Authorize.net, CyberCash, or any other secure order provider with a shopping cart. Another feature is the Auction support for Ebay.com with one-click auction listing capability. The Cart is also integrated with the Customer Support HumanClick and IcQ putting the customer in immediate contact with Customer Service as they shop and learn about the products.

Store HTML templates can be edited with FrontPage, PageMill or any other HTML editor. There are no limits on design capability, number of products, number of departments or number of options per product. This software runs on UNIX, Linux or Windows based server and uses database to store product information.

8.3.3 Catalog Management systems

Catalog management systems are used to store, retrieve, and display the information about products and services that are sold in EC. A catalog can be a simple relational database that keeps product information or a special purpose catalog system that stores, in addition to data, textual and graphical information. Examples of few specialized catalog systems are briefly discussed below.

POET eCatalog Suite™ addresses the market need to automate the supplier’s creation and interchange of catalog data with their customers and business partners. Each customer has unique

needs, i.e., product needs, negotiated pricing, processes and computer systems. Hence customers represent an unlimited number of permutations of any supplier's catalog data. The POET eCatalog Suite complements the supplier's existing IS infrastructure, providing catalog information about goods and services that can be customized for each customer's specific requirements. It would be nearly impossible to manually generate all the different versions of a catalog to address all of these permutations on a mass scale. POET addresses this situation by extracting the data from your existing systems, and storing it in a rich generic data store. When a customer requests a catalog, the information is pulled from the master catalog and customized via a customer profile to produce a catalog that is completely customized for that customer. This includes special pricing, data format, data exchange mechanism, product/service selection, and more. As a result, suppliers are in a position to ensure their buyer receives exactly the catalog they need to successfully conduct electronic commerce. POET claims to have effectively exploited the Internet to provide universal connectivity, mass customization and improved customer relationships providing B2B suppliers to easily assemble their catalog data into a single master catalog from which custom catalogs are generated on the fly.

IBM Catalog Architect is another suite of products designed for businesses that use IBM Net.Commerce Hosting Server merchant server software. It enables on-line businesses to create, update, and manage product information, providing a high degree of efficiency, accuracy and detail while reducing time spent on traditional catalog information management. Catalog Architect is designed to understand the inherent relationships between different catalog elements, such as products, categories, product sets or kits, SKUs or items, and cross-sell items. IBM Catalog Architect, too, enables personalization and advanced catalog searching, simplified creation and management of information-rich electronic catalogs, providing a familiar spreadsheet, with drag and drop interfaces that require no special database, and cut-and-paste capabilities for multiple attributes, products and categories. It provides intelligent catalog searches to guide customers to product selection.

IBM Net.Commerce Product Advisor, part of the suite, can provide detailed parametric searches, virtual sales assistance, and product comparisons. Its object-centered constraint model architecture provides the foundation for the catalog information knowledge base. By eliminating redundant product information and allowing a single point of entry for modifying multiple products and SKUs. IBM Catalog Architect enables the merchant to automatically create new items or SKUs by inheriting all the previously existing product information, as well as the new attribute values, significantly reducing the time spent on data entry and content management. It allows businesses to import or to export information from or to IBM Catalog Architect in an XML format.

There are also available more user-friendly, no-experience-needed catalog products providing 'store building wizards' to allow merchants to build and manage catalogs, and arrange for payment processing methods. iCAT Web Store enables a merchant to build a customized, browser-based catalog, not requiring any downloading or special hardware, using pre-designed templates. It provides a full range of merchandising and marketing techniques, payment processing and security, catalog management and updating, business reports, e.g., customer, order and tracking reports, 24 **x** 7 maintenance for hardware, as well as, assorted merchant education features.

A design and hosting service available is WebCog Commerce, offered by Turnaround Computing. They work with the merchant to map out the catalog, design the web pages, determine the inventory and vendor policies, establish payment verification, gather images, establish customer and administration policies, and test and promote the web site. Requiring somewhat more sophistication on the part of the merchant, but still not requiring any programming, the merchant uses HTML to update his catalog.

8.3.4 XML for Ecommerce

Most enterprise software vendors at present support XML for C2B trade, and standards are being defined to further simplify the interchange of data using XML. As discussed in a previous chapter, XML is a markup language, similar to HTML, for documents containing structured information. XML can be used easily to represent information to be exchanged between traders. Consider, for example, the following XML document: that represents customer information:

```
<?xml version="1.0" standalone="yes"?>
< -- customer example -- >
<customer>
            <name>
                  <first>Pat</first>
                  <last>Hemsath</last>
            </name>
            <address>
                  <street>room 225</street>
                  <street>410 Hoes Lane</street>
                  <city>Piscataway</city><state>NJ</state>
                  <zip>08854</zip>
            </address>
            <phone>732-699-1111</phone>
</customer>
```

A Document Type Declarations (DTD) specifies a set of rules for the structure of the document and can be used to verify the document. For example, the following is a DTD for the customer record just defined and can be used to verify the customer information: :

```
<!ELEMENT customer (name, address?, phone?)>
<!ATTLIST customer id CDATA #REQUIRED>
<!ELEMENT name (first, middle?, last)>
<!ELEMENT address (street+, city, state, zip)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```

This DTD specifies the valid tags to be used jn defining a customer record and can be used to eliminate invalid records in EC. Figure 8-6 shows the role of XML and DTD in B2C EC. In this case, a business creates a specification of, say, a purchase order (PO) that is stored in a repository. This specification represented as a DTD and is downloaded by the consumers for submitting to the businesses. The XML repository (DTD) may reside at the host business or at a separate site. The consumers (buyers) download the PO format and create POs in XML. The submitted PO are sent to the business (seller) and the the XML gateways at the business provide the XML parsing and verification against the DTD. The invalid submissions are discarded and the valid ones are forwarded for processing. Similar gateways may exist at the consumer sites also. The next section shows a sample PO in XML and a corresponding DTD.

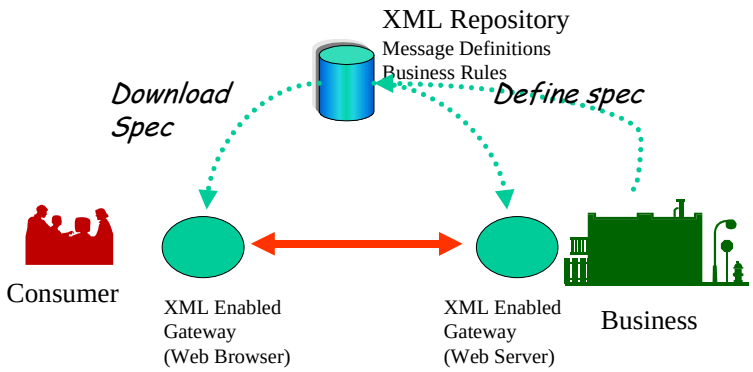


Figure 8-6: XML in C2B Ecommerce

However, adoption of XML for EC is not risk free.. Unfortunately, there are many XML standards for different industry sectors, and even several within the same sector. The possible ways forward are:

- Adopt super standards repository "framework" such as the **Microsoft's BizTalk** (http://www.BizTalk.org). The BizTalk framework aims to make it easier for individual companies to mix and match XML message formats from different vendors and standards groupings, picking out the sets that best meet their business needs and application mix. The basic idea is that your company subscribes to BizTalk-conformant standard A, i.e., you build interfaces by using standard A. Your business partner, let us assume, subscribes to a different standard, B, which is also Biztalk-conformant. Using the XSL translation between A and B (available from the Biztalk repository), you can send XML messages in standard A, which your partner can then translate from A to B and understand your messages. The success of this depends on the acceptance of Biztalk and the ability of Biztalk partners to control complexity.
- Manage the XML interfaces properly within your own company. You can build a single technology-independent logical model of the information needed to drive *your* business. You can then map all the different technology pieces onto that logical model. Any data translation between trading partners is not done directly, but in two steps via the logical business model. This can help you control the complexity and proliferation of XML.

8.3.5 Sample XML Source and DTD for Purchase Order

The following statement represents XMLSource for a purchase order:

```
<?xml version="1.0" encoding="UTF-8"?>
<PO>
        <POHeader
                <description>software program</description>
                <paymenttype>Visa</paymenttype>
                <shiptype>UPS</shiptype>
                <fromcust>Zombie Jr</fromcust>
                <PONumber>12567</PONumber>
        </POHeader>

        <Contact>
```

```
            <contactname>Jim Shorts</contactname>
            <contactemail>jimshorts@jimbojerky.com</contactemail>
            <contactphone>212-EAT-JERK</contactphone>
        </Contact>

        <POShipTo>
            <city>New Brunswick</city>
            <attn>CIO</attn>
            <country>USA</country>
            <stateprovince>NJ</stateprovince>
            <street>Anystreet</street>
            <zip>11234</zip>
        </POShipTo>

        <POBillTo>
            <city>New Brunswick</city>
            <attn>CIO</attn>
            <country>USA</country>
            <stateprovince>NJ</stateprovince>
            <street>Anystreet</street>
            <zip>11234</zip>
        </POBillTo>

        <Item>
            <unitprice>$129.99</unitprice>
            <qty>1</qty>
            <inventorynum>37893</inventorynum>
            <needafter>August 29, 2001</needafter>
            <discount>.20</discount>
            <needbefore>September 30, 2001</needbefore>
        </Item>
</PO>
```

The following DTD describes this source purchase order:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PO[
<!ELEMENT PO(POHeader,Contact,POShipTo,POBillTo,Item)>

<!ELEMENT POHeader(description,paymenttype,shiptype,fromcust,PONumber>
<!ATTLIST POHeader
        paymenttype (Visa|MasterCard|AMEX) #REQUIRED>
        shiptype (UPS|FedEx|USPS) #REQUIRED>
        PONumber ID CDATA #REQUIRED>
<!ELEMENT description (#PCDATA)>
<!ELEMENT fromcust (#PCDATA)>

<!ELEMENT Contact(contactname,contactemail,contactphone)>
<!ATTLIST Contact
        contactname CDATA #REQUIRED>
        contactemail CDATA #REQUIRED>
```

```
        contactphone CDATA #REQUIRED>

<!ELEMENT POShipTo(city,attn,country,stateprovince,street,zip)>
<!ELEMENT POBillTo(city,attn,country,stateprovince,street,zip)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT attn (#PCDATA)>
<!ELEMENT  country (#PCDATA)>
<!ELEMENT stateprovince (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT zip (#PCDATA)>


<!ELEMENT Item(unitprice,qty,inventorynum,needafter,discount,needbefore)>
<!ATTLIST Item inventorynum id CDATA #REQUIRED>
<!ELEMENT unitprice (#PCDATA)>
<!ELEMENT qty (#PCDATA)>
<!ELEMENT needafter (#PCDATA)>
<!ELEMENT discount (#PCDATA)>
<!ELEMENT needbefore (#PCDATA)>
]>
```

XML Resources

Information about XML is growing rapidly. Here is a very small list to get started:

- Extensible Markup Language (XML) 1.0 description at http://www.w3.org/TR/WD-xml
- www.xml.com, a very useful Web site for a great deal of information about XML
- ebXML initiative and the XML/EDI group Web sites
- "Xml by Example : Building E-Commerce Applications", Book by Charles F. Goldfarb Series on Open Information Management) by Sean McGrath (Paperback, latest edition)
- "Xml for EDI : Making E-Commerce a Reality" Book by Hussain Chinoy, et al (latest edition)
- www.w3schools.com, a very good source for great tutorials

8.3.6 Ecommerce Transaction Processing

Business transactions are at the core of electronic commerce. Examples of typical EC transactions are purchasing, claim processing, and billing/payment. For business to business activities in ECs, the importance of supporting highly reliable and secure business transactions is quite obvious. Formally, a *transaction* is a collection of operations on a database which has the so-called ACID properties (we take a closer look at ACID properties in Section 8.10.1.1):

- **Atomicity**: All of the operations in the transaction must take place, or none must take place. In practice, if any of the elementary steps that are part of the transaction action fails, then all the steps must be undone;
- **Consistency**: The result of performing all the operations in the transaction is to take the database from one consistent state to another consistent state;
- **Isolation**: Other users of the database are isolated from any intermediate states of the transaction, i.e., they may see the state of the database before the transaction begins or after it completed, but not any state in the middle;

- **Durability**: Once all the actions in the collection have completed, the effects endure even in the event of system crashes.

EC is a mixture of decision support and transaction processing activities. Normally, only a portion of the core EC activities are transactional. The ACID requirements for these activities are analyzed in Table 8-1. Two main types of EC transactions are relevant:

- EC transactions between trusted business partners (e.g., suppliers and corporations that enter business agreements and contracts to buy and sell products). These transactions typically are large in volume (large amounts of money and goods), introduce medium traffic, and require rigorous security. These transactions are perfect candidates for Extranets (see Section 8.3.1);
- EC transactions between suppliers and the general public (e.g., Internet shopping malls).

Transactional support is implemented differently in different types of systems. For centralized mainframe systems, on-line transaction processing (OLTP), has been built for ACID transactions and has been a backbone of commercial data processing since the early 1970s. Mainframe-based transaction managers (TMs) such as CICS and IMS-DC/IMS-TM have matured over the years to provide high performance and reliable services The situation is dramatically different in distributed environments that characterize EC. In these environments, the approaches fall into the following categories (see the discussion on Distributed Transaction Management in Section 8.10 for a more detailed discussion):

- TP-Less, i.e., do not use any transaction management facilities;
- TP-Lite, i.e., use database procedures to handle updates;
- TP-Heavy, i.e., use a distributed transaction manager to handle updates.

Which of these approaches works depends on the type of EC activities being considered. It appears that each one of these approaches have certain pluses and minuses for EC. The following questions should be asked before deciding on the approach:

- In what format is the data stored (databases, flat files)? If the data is stored in multiple databases and flat files, then TP-Lite is not suitable (database procedures only work in RDBMS environments);
- How many SQL servers does the data reside on? If the application needs to update and commit data that is stored on multiple servers, then TP-Heavy should be used (database procedures cannot participate with other database procedures in a distributed transaction).
- What is the requirement for data synchronization? If the data synchronization interval is periodic, then a TP-Lite solution combined with a data replication server may be useful to handle updates against replicated data.
- What are the requirements for performance and load balancing? TP-Less works well when you do not need any transaction processing capabilities. TP-Lite solutions with database procedures are much faster, on the surface, than the TP-Heavy solutions that require synchronization between sites. But TP-Heavy solutions provide many sophisticated procedures for dynamic load balancing, priority scheduling, process restarts, and pre-started servers that are especially useful for large scale production environment. These features are the main strength of TP-Heavy products because many of these products have been used over the years to handle thousands of transactions in production OLTP (on-line transaction processing) environments.
- While the debate between the TP-Lite and TP-Heavy proponents continues, most EC projects are completely ignoring this whole issue by focusing primarily on EDI and in some sense re-inventing the wheel. EDI, at best, is a TP-Less approach. Some EC applications are being deployed by using TP-Lite while large mission critical EC applications, if any, use TP-Heavy only at back-end mainframe systems. In the meantime, it seems that many small EC applications are quite happy with TP-Less.

The following areas are of particular interest in transaction middleware for EC:

- Transaction processing with distributed objects, concerned with performing ACID operations between objects across machines, introduces many complex issues. The Object Management Group (OMG) has did the initial work on transaction processing with distributed objects as a service for its Common Object Broker Architecture (CORBA). At present, most of the imilight in this area has shifted to Web services.
- Transaction processing with object-oriented databases is concerned with introducing TP Heavy type of operations on top of OODBMSs. Most of the TP Heavy type of operations are currently available on relational and hierarchical databases. The fundamental difference is that unlike older DBMSs that provide read and write operations, the OODBMSs support methods that are rich in semantics. Most of the work in OODBMS has concentrated on query and modeling aspects. In particular, transaction management in object-oriented multidatabase systems is receiving serious attention recently. We will have to see how this technology finds its way in EC. Since OODBMSs are not common in EC (no surprise here), we will not discuss this subject any further in this paper. In EC, OODBMSs can be used as "transaction databases" (TDBs) for storing shopping carts, purchase orders, payments, income receipts, and invoices. In addition, customer and inventory information are stored in databases to support the order processing activities. These transaction databases are quickly becoming the foundation of transaction processing and workflows in EC.
- Integrated End-To-End Business Transactions. This middleware will extend and meld the current generation of EDI, workflow, and distributed transactions for an end-to-end reliable and secure business transactions. For example, a workflow system at COI site A will generate an EDI transaction that will be received by the EDI systems and workflow systems at sites B and C for automated processing.

Distributed transaction, the core of ecommerce systems, has been an active area of research and development since the 1970s. A very brief overview of the technical issues involved in distributed transactions is given in Section 8.10.

EC Activities	Atomicity	Consistency	Isolation	Durability
Advertising	Not needed	Not needed	Not needed	Not needed
Browsing and Selection	May be needed for purchase carts	Not needed	Not needed	Not needed
Purchasing	Needed	Needed	Needed	Needed
Billing	Needed	Not needed	Not needed	Needed
Payments	Strongly needed	Needed	Needed	Needed

Table 8-1. Is ACID support needed for EC?

.

Internet Transaction Processing (ITP) – Making Transactions "Internet Aware"

Transaction processing (TP) in Internet is the backbone of electronic commerce because all buying/selling activities involve transactions. However, Internet (unsupervised and free access) is

contrary to the notion of commercial transactions (highly controlled and supervised). In particular, TP requires the following:

- Guaranteed message delivery, i.e., once you have sent a message the middleware will be responsible for delivering the message. If the message is not delivered, the middleware will try to deliver it again.
- Guaranteed message processing, i.e., once the message is delivered, it is processed correctly (committed). If a message processing error occurs, the results of processing are rolled back.
- Improved security and privacy i.e., the transactions are protected from unauthorized users. The transaction activities are also logged for audit trails.

How are these requirements being satisfied in the Internet? First, increased use of message-oriented-middleware (MOM) provides guaranteed message delivery. Second, Internet TP typically uses the fat server model in the Internet world because most of the TP processing continues to take place in the back-end systems with proven TP technology for guaranteed message processing. The client side may include Java applets that do minimal processing. Third, the servers reside inside the corporate firewalls for security and the public clients reside outside the firewalls. The clients and the servers use improved security services such as public/private keys and Secure Socket Layer (SSL) in the Internet world. These factors, plus consortia such as CommerceNet, will be vital for electronic commerce. Internet transactions are typically supported by the e-commerce platforms.

Naturally, Internet Transaction processing (ITP) involves many issues. For example, ITP must deal with the philosophical differences between running transactions on the Internet, as opposed to the corporate private network. While the Internet is considered unsecure, OLTPs are private and controlled by corporations. To strike a balance, the corporations are using the firewall as a basis for partitioning the ITP services. For example, clients use the Public Internet that is outside the firewall to submit requests and receive answers. The actual transaction processing is performed inside the firewall by the application services with the assurances of protection and integrity controls typically needed for transaction processing.

Tandem introduced iTP (Internet Transaction Processing) in October 1996 to extend its OLTP products into the electronic commerce arena. The iTP strategy consists of six servers: Tandem iTP Commerce Server (which enables business-to-commerce transactions on the Internet), Tandem iTP Media Server (for the distribution of videos, electronic catalogs, music, and large database content over the Internet), Tandem iTP Messaging Server (it enables businesses to communicate with each other through X.400 and SMTP), Tandem ITP CTI Server (to support computer telephony integrated applications), Tandem iTP Intranet Server (it facilitates collaboration by geographically distant team members), and the Tandem iTP Matrix Server (it allows integration of legacy and other custom applications). In addition, Progress Software Corporation has also built an Internet Transaction Processing architecture. Numerous other attempts have been announced and are on the drawing board.

8.3.7 Electronic Payment Systems -- An Example of Transaction Processing

8.3.7.1 Overview

Electronic payment systems are central to EC because on-line consumers must pay for products and services. In particular, payments and settlements must be resolved between all partners (customers, merchants, banks, brokers, etc) quickly and smoothly otherwise the whole business chain is disrupted.

On-line sellers need to decide how best to support payments for on-line purchases, what type of currency to use, and what type of electronic fund transfer (EFT) system to use. Electronic payment systems are the best known examples of transaction processing. The payment is typically handled by a **Payment Server,** sometimes also known as **Commerce Server.** Current approaches to payment fall into the following broad categories:

- Retailing payments such as credit cards (e.g., VISA or MasterCard), charge cards (e.g., American Express), or private label cards (e.g., Sears cards).
- Banking and financial payments such as large scale or wholesale payments (e.g., bank-to-bank transfer), retail or small scale payments (e.g., cash, ATM cards, checks)
- Digital token-based systems that include electronic cash, electronic checks, and smart cards

The first two categories (retailing and banking systems) are not completely adequate for large scale EC -- they assume that the parties will be in physical presence and enough delays will be built into the system for frauds and overdrafts to be detected. Most of the current work in EC payments concentrates on different types of token-based systems. Most common examples of token-based payment systems are:

- **Electronic cash (e-cash).** This method involves "digital signatures" that enforce public/private keys (to be explained later) to identify buyers/sellers. In practice, the buyer establishes an account with a bank and is responsible to keep enough money in the account to pay for the purchases. The customer is issued a token for, say $100, and the token value is reduced every time a purchase is made.
- **Electronic checks.** This is essentially an automation of paper checks. You open a checking account and then send your checks through email to the seller. The seller sends this check to the bank through an accounting server that performs various authentications before clearing the check.
- **Smart cards.** These are basically credit cards with microprocessor chips that can hold much more information than the magnetic stripes of the traditional credit cards. In some cases, the smart cards enhance the services of traditional cards (traditional cards are evolving into smart cards that allow processing from multiple accounts) or provide electronic purses (supply quick electronic cash for buying soda and candy). Smart cards need special smart-card readers.

A variety of token-based payment systems for EC exist that combine various features of e-cash, e-checks and smart cards. Examples of such systems are:

- **Cybercash.** This system uses encrypted method for shopping on the internet and supports an Internet wallet that holds credit cards and electronic checks. Cybercash provides Cybercoin to pay for small ticket items (25 cents to $10). Cybercash is in partnership with Netscape for these services.
- **Verifone.** Verifone is a leader in POS (point of sale) terminals. Verifone supports vWallet for electronic purchasing and is in partnership with Microsoft (Merchant Server)
- **GCTech.** GCTech is supported by Global Online Corporation, based in France. This system interfaces with numerous credit cards and banks for payment.
- **SET (Secure Electronic Transactions).** SET is a potocol for credit card transactions over the Internet. It deals with authorization, settlement, etc. Development of SET has been led by Visa and Mastercard.
- **NetBill.** NetBill was developed at Carnegie Mellon for low cost, high volume transactions. The system was initially used to purchase information from digital libraries. A partnership between Visa and Carnegie Mellon has been formed for pilots.

The following paragraphs will bridge the gap between the customer clicking on the purchase button to the "monthly check" sent to the on-line merchant.

8.3.7.2 Merchant Account Providers (MAP)

A number of different web-sites use the label of Merchant Account Provider, or MAP, to describe themselves. The term MAP refers to any service that will **verify** the credit card, **process the transaction**, and **deposit** the results into your **account**. The term is often used loosely, and can refer to providers of merchant accounts without payment processing, (banks); providers of merchant accounts and on-line credit card processing, such as independent sales organizations (ISOs); and providers of on-line credit card processing that refer customers to merchant account providers on request, such as credit card processors.

The terms MAP, ISO and merchant service provider (MSP) often are used interchangeably. However, while the terms are similar, services and fees among these providers can vary considerably.

8.3.7.3 Banks

Banks are viewed as the most secure and reliable option, but they are also the most selective. For an established low risk business it should be easy to open a merchant account in a bank. The bank can usually arrange a third-party processor to set up a mechanism for accepting credit card payments. U.S. Bank, for example, uses CyberCash (www.cybercash.com) a company that offers Web-based payment-processing software. The merchant downloads this software directly from CyberCash, and is given the option to buy the software or lease it on a monthly basis. This is typical of many bank merchant account arrangements.

8.3.7.4 Independent Sales Organizations (ISOs)

Most ISOs offer merchant accounts and the ability to process on-line credit card transactions in exchange for a transaction fee and a percentage of sales. Unlike banks, ISOs are generally more tolerant of high-risk accounts because they are not monitored or as tightly regulated. In fact, much of their business comes from companies that cannot obtain merchant accounts from banks directly.

8.3.7.5 Credit Card Processors

These companies are responsible for processing the credit card transactions -- verifying, approving and then transferring funds securely from one bank to another. They are not considered MAPs per se, as they do not provide merchant accounts. Instead, they form relationships with banks and ISOs to integrate payment processing with merchant accounts. For example, many ISOs use the services of the credit card processors CyberCash and Authorize.NET.

There are three ways to handle credit card transactions:

- **Terminal Processing** -This method is most suitable for retail stores or businesses that will have to access the physical credit card of their customer to swipe it on a terminal machine. It is quick and efficient. All that is required is a swipe of the credit card through the machine and it is completed.
- **Software Processing** - The least popular method of processing payments is with a computer software program. It is very similar to a terminal, but requires the operator to enter the customers information, and credit card number to process the order. Although it is not possible to swipe the card, and the numbers must be entered into the software that is installed on the computer, it is possible nevertheless to store all recurring customers' information in the program for easy billing.

- **Real-Time Processing** - The most popular method of accepting credit cards for businesses on the web is real-time processing. This type of merchant account will process and charge your customers' credit card information automatically by working with the chosen ordering system (Shopping Cart, Order Forms, etc.). Since the whole process is automatic, this is the most effortless method available.

The simplest method to handle Credit Card (CC) transactions is to manually verify each credit card payment through a terminal or software , then pack and ship the goods. If customers are unwilling to wait for manual verification, or your sales volume is high, a real-time card verification software is needed. One such tool, ICVerify[i], is available for DOS, Windows, and Unix, and is integrated with many Web storefront packages. ICVerify collects credit-card information from a Web form and processes the authorization in real time, depositing the purchase price amount in your merchant account. ICVerify currently requires a dedicated connection to a bank or credit-card processing company, although an Internet-based solution called WebAuthorize is available from CyberCash Inc.

Secure Electronic Transaction (SET). An alternative credit-card processing scheme, supported by card-issuing banks, is the Secure Electronic Transaction (SET) protocol developed by Visa and MasterCard, and now backed by American Express. Designed for cardholders, merchants, banks and other card processors, SET uses digital certificates to ensure the identities of all parties involved in a purchase. SET also encrypts credit and purchase information before transmission on the Internet. SET is designed to protect the transfer of bankcard payment information over open networks like the Internet. An application layer security protocol, SET describes the dance between four players: consumers, merchants, merchant bank, and consumer bank (Figure 8-7). From an end-user's point of view, the purchase transaction is separated into two parts by SET:

- Purchase information that is sent to the merchant
- Credit card information that is only handled for credit card verification. This information is not sent to the merchant

8.3.7.6 Merchant Service Providers (MSPs)

The term MSP refers to banks, ISOs, or other institutions that offer financial transaction processing, usually related to credit card sales. Many MSPs provide merchant accounts; others require customers to establish them independently.

8.3.7.7 Third party Payment Processors.

Services such as iBill, CCNow and Verza allow merchants to accept credit cards without a merchant account. Instead, transactions are processed through the third-party providers' accounts (MAPs). These companies charge a processing fee with each transaction that's higher than the discount rate charged by most ISOs. However, this is the only fee the merchant incurs. This makes third-party processing services perfect for small businesses just getting started that lack established credit and generate less than $1,000 per month (U.S.) in sales. These services are also popular with older businesses that have poor credit, with non-U.S. businesses and with businesses selling content and services.

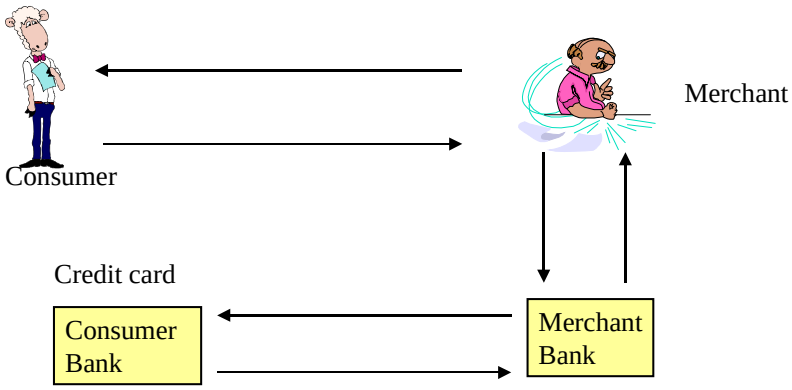


Figure 8-7: SET Processing

Time to Take a Break

- Overview & eCommerce Middleware
- eCommerce Security
- eCommmerce Platforms
- Distributed Transaction Management

Suggested Review Questions Before Proceeding

- What type of middleware services are needed to support e-commerce?
- What role does XML play in e-commerce?
- List the middleware services needed to support online magazine subscriptions.
- Compare and contrast transaction processing with payment systems.

8.4 Security for e-Commerce/e-Business

8.4.1 Overview

The issues of security are of vital importance for EC/EB and need more attention. Basically, security involves the following aspects:

- **Privacy and Integrity:** assure privacy of information (i.e., no one other than the authorized people can see the information) when transmitting it over a network or storing it in a insecure place. In addition, the integrity of information (i.e., no unauthorized modification) must also be maintained.
- **Authentication:** identify for certain who is communicating with you (i.e., make sure that you are who you are)

- **Authorization (Access control):** determine what access rights that person has (i.e., can you only read given information or can you also update, delete, add information).
- **Accountability and Assurance:** assure that you can tell who did what and when and convince yourself that the system keeps its security promises. This includes ***non-repudiation (NR)*** -- the ability to provide proof of the origin or delivery of data. NR protects the sender against a false denial by the recipient that the data has been received. It also protects the recipient against false denial by the sender that the data has been sent. In other words, a receiver cannot say that he/she never received the data or the sender cannot say that he/she never sent any data.

You also need to administer the security system, i.e., define and enforce the security policies that are consistent across all elements of applications, middleware services, and networks. These, and other aspects of security, are supported at various levels (network, middleware, application) by using a wide range of technologies (see Figure 8-8). Security is needed at these different levels since security at each level fulfills different requirements. Let us briefly review the security at various levels (details will be given later).

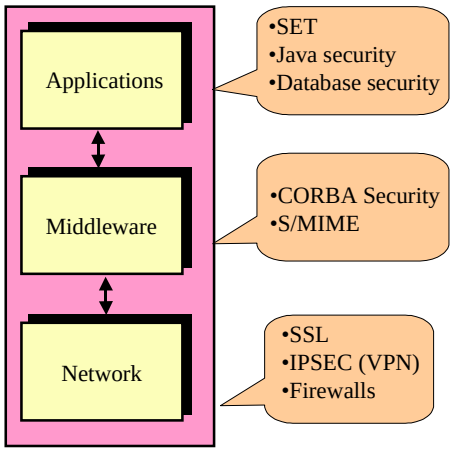


Figure 8-8: Levels of Security

Network security protects communication and transactions data. "Firewalls" and "gateways" are erected to regulate traffic. In addition, the network traffic can be encrypted at packet level (IPSec) or at the transport level (SSL- Secure Socket Layer). Middleware can also imbed security. CORBA security is a good example that assures that CORBA applications are secure. In addition, S-MIME secures email. A variety of security approaches exist at the application level, in which case authorization controls are used within applications to regulate access to specific data, and cryptographic infrastructures are built to strongly authenticate users and provide confidentiality. Examples of application level security is provided by database managers, Java security, and SET (Secure Electronic Transactions). In particular, applications themselves provide access control and strong user authentication.

Security must be considered at all levels. Securing a higher layer while keeping lower layers unsecured makes the system vulnerable to intrusions from the lower layers. In general, lack of security at a certain layer might compromise the overall system even if other layers are secured. Consider, for instance, a system where the application data is secure, but is transmitted over an insecure network. In this case, the overall security of the application could be suspect. Specifically, application security protects application data (e.g., database security mechanisms allow the data to be stored on the hosts in a protected manner) and system resources (e.g., Java Security) while SSL protects data while being transferred on the network.

A ***security design approach*** is needed to include various security issues at different levels. Basically, it is important that the business logic of a Web application runs on a server and not on the client. The Web application server can be used to integrate access to resources (databases, etc.), which provides greater security of the resources. In addition, you should structure your application by using network filters ("firewalls"). A good design protects the Web server (providing presentation services) behind an outer firewall, and the remaining servers (supporting business logic) behind a second, inner firewall. This structure, shown in Figure 8-9, is known as a demilitarized zone, or DMZ. In most cases, a Web server sits alone in the DMZ, handling requests from the Web and passing them along to the secure intranet network. The applications and internal business systems behind the inner firewall contain all the remaining business logic and data of the application. In addition, you can gain performance benefits by caching frequently requested data inside the DMZ rather than retrieving it from back-end systems each time it is requested. However, machines in the DMZ are known to be at higher risk. In addition to DMZ, you also need to consider security of clients. For example, mobile devices typically need another level of security before they can enter the DMZ.

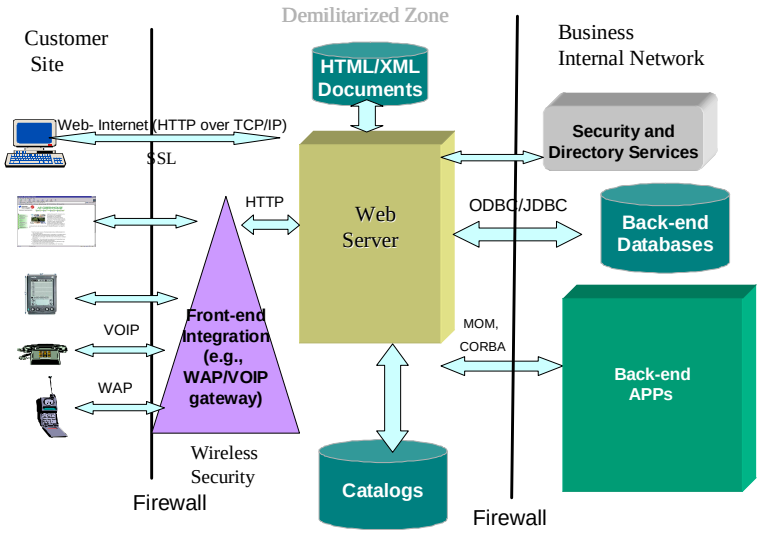


Figure 8-9: Security Design

The balance of this section gives a quick overview of security technologies and presents an illustrative example to highlight the key issues.

8.4.2 Overview of Core Security Technologies

User logon and password is one of the oldest and still most commonly used technology. In this case, a system keeps track of who can access that system. This technology enables the use of existing systems with minimal disruption to existing infrastructure and applications.

Encryption is another technology that has been used for a number of years to mask the messages so that the interveners cannot see/modify the messages. Due to e-commerce, encryption/decryption has become a major area of active work. In the simplest case, data is transformed by a key into an encrypted message. The encrypted message is then transmitted and decrypted on the other side by using the same key. Encryption/decryption can be performed by hardware and/or software. Modern computing systems have the ability to implement very sophisticated encryption/decryption techniques. The same encryption can be used on all data in a system or encryption keys can be more "personalized". For example, instead of using the same encryption/decryption key on all data from all

stations in a network, each station or user can use its own encryption/decryption key. A user can have his or her own encryption card which is inserted into a workstation before the user logs on. This card encrypts the data before sending it across the network. The encrypted data can be read only by those users or programs with access to the same encryption key. Encryption is generally discussed in two different formats:

- **Secret Key**: In a secret key (also known as symmetric or private) encryption scheme, the same key is used by the sender to encrypt the message and the receiver to decrypt it. While secret key encryption is usually very fast and efficient, the problem is with key management. In other words, since the sender and receiver have to agree on the same key, sending the key from one side to the other might compromise it.
- **Public key**: In a public key (asymmetric) system, the encryption key E and the decryption key D are different - hence the name "asymmetric". Each user has a pair of keys, a private key D that he keeps secret and a public key E that he publishes. When sender Bob needs to send a message to a user Joe, he encrypts the message with Joe's public key E(J) . This encrypted message can only be decrypted with Joe's private key D(J). Therefore if this encrypted message is delivered to user Pat who does not have key D(J), then Pat cannot decrypt it. Thus when Joe receives the message, he can decrypt the message by using D(J) and read the message. Notice that in this key system the decryption key is private and not transmitted over the network. While public key systems solve the problem of key management, they are usually significantly slower than private key systems. The RSA (Rivest, Shamir and Adleman) algorithm, developed in 1976 is by far the most wildly used public key encryption algorithm.

Digital signature is used to authenticate the source of a message. It is essentially the same as a public key system except that the order in which the keys are applied is reversed. A sender “signs” the message by applying his private key to it. The sender sends the message and the signature to the receiver. The receiver checks the signature by applying the sender’s public key to it. If the receiver gets the original message back, he is sure that the message was signed by the sender’s private key, and therefore, was sent by the receiver himself. In essence, a digital signature is a block of data created by applying a cryptographic signing algorithm to some data using the signer's private key. Digital signatures may be used to authenticate the source of the message and to assure message recipients that no one has tampered with a message since the time it was sent by the signer

Message Digesting is used to make sure that a certain message was not changed along the way between the sender and the receiver. A message digest algorithm produces a fingerprint of the message, by applying a hashing function to it. The receiver can check for the integrity of the message by reapplying the hash function and comparing with the original fingerprint. The hash functions used in these schemes are such that the fingerprint changes dramatically if a single bit of the message changes.

A digital certificate binds an entity's identification to its public key and is issued by the Certification Authority. Digital certificates, based on the X.509v3 standard, enable Internet applications and other users to verify the identity of an entity. Unfortunately, certificates produced by one vendor product may not interoperate with other vendor's because X.509 does not define the formats of the certificate entries and other necessary provisions. PKIX, the X.509 standard by IETF, [who’s this ?] defines the contents of public key certificates and is intended to resolve these interoperation issues.

8.4.3 Information Protection (Privacy and Integrity)

Information must be protected and its integrity maintained at least at two levels: a) the sites where it exists, and b) when it is transmitted. In addition, the encryption keys themselves need to be protected.

Site Protection. Information must be protected at the sites where it exists. Access control (allowing authorized users to access needed data) protects data at various sites. Most database managers have security features that allow only authorized users to access needed data. In some cases, data is encrypted and stored for additional security. An important aspect of site protection at present is **Java Security.** Security of Java code has been an area of concern for a while. The current Java security is defined at the following levels:

- **Java1.1's security management system --** All local code is trusted. All remote code is untrusted, unless it is digitally signed by a trusted source. Untrusted code runs in a "sandbox", and has limited access to local system resources.
- **Java 2's security management system --** Local and remote code are checked by the same security management system. It supports fine-grained, flexible and easy-to-specify security and permission policies.

b) Transmission Protection. When data must travel outside of a secure system environment, it needs to be protected so that the policies governing its use cannot be violated. Secure communications, ensuring data privacy, data integrity, and origin authentication are an important aspect of information protection. Examples of the technologies used for secure communications are:

- **Firewalls** -- the network filters that police "who" enters and leaves an enterprise network and "what" gets in and out. A firewall is essentially a software package that is installed on network routers. This software checks each IP packet and determines if it should enter the system. Firewalls provide a logical and physical separation of the public Internet and internal IT systems. A good security design generally has two firewalls: an outer firewall that exposes some services to the outside world and a second, inner firewall, that keeps the inner resources. The zone between the two firewalls is known as a demilitarized zone, or DMZ.
- **SSL** - The Secure Sockets Layer (SSL) protocol uses encryption and authentication techniques to ensure communications between a client and a server remain private and to allow the client to identify the server and vice versa. SSL runs on top of TCP/IP and manages secure messaging on the network. SSL client and server negotiate encryption scheme and key size. SSL is currently used heavily to protect the traffic between Web clients and servers. It uses RSA (Rivest, Shamir, and Adleman) Public encryption for key session negotiation and DSA (Digital Signature Algorithm) for session encryption. See the sidebar "SSL" for additional information.
- **VPN and IPSec**- Virtual Private Networks (VPN) are private networks (e.g., networks internal to corporations) that use public communication infrastructure. In other words, you set up a private network over a public network by using encryption. VPNs use IETF IPSec (RFC 2401) and related standards to transport encrypted messages over shared networks. IPSec provides security at the packet level, instead of security at application layer. It encrypts and signs Headers and/or Data parts of the IP Header. It provides security without requiring changes to applications and thus is suitable for Virtual Private Networks (VPN). VPN differs from SSL in that it creates a secure channel between two TCP/IP hosts over which multiple TCP/IP connections can be established. Each TCP/IP session itself may or may not use SSL. See Section 8.3.1 for additional discussion of VPN.
- **S/MIME** - Most e-mail client and server programs using Internet systems such as SMTP send e-mail as clear text. The Secure Multipurpose Internet Mail Extensions (S/MIME), a specification for secure electronic messaging, can be used to prevent the interception and or forgery of e-mail.
- **Middleware Security** - Web security is a good example of protection at middleware level. This security can be divided in terms of:
  - Web client security
  - Web Server security
- **SET** - The Secure Electronic Transaction (SET) protocol, developed jointly by Visa, MasterCard, IBM, and other technology providers, is used to protect the transfer of bankcard

payment information over open networks like the Internet. This is an application layer security protocol.

c) Key Protection. In addition to secure communications, protection of the keys that in turn are used to protect the assets is also important. Private keys and shared secrets, once acquired, must be protected. End-to-end security must include consideration of the security of the end user device. Private keys stored on a personal computer disk file may be stolen via access to the file system or outright theft of the device. Security can be enhanced by the use of smart cards. Another approach is to use a security chip embedded in end user systems. In addition, server-side hardware devices can provide tamper resistant key storage as well as assistance for encrypting and decrypting messages and public/private key operations, etc. that require heavy computational load.

Basic Security Services: SSL and Digital Cetificates

At present, most Web browsers and servers use Secure Sockets Layer (SSL) technology to provide a safe way to transmit sensitive information, such as credit card numbers, on-line banking, email messages, surveys and other personal information.

The SSL protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection, allowing for the secure transfer of sensitive information over the Internet. SSL consists of software installed in browsers and on servers and can be obtained by subscribing to a Secured Service Provider such as ssl.com or by obtaining a Server Certificate from ssl.com and installing it on an existing secured server.

All major browsers and servers today are "SSL capable". SSL technology was developed by Netscape Communications Corporation and has become the industry-standard method for protecting web communications.

SSL Overview.

SSL uses public key encryption to provide security at the packet level. At the receiving end, SSL provides Server Authentication Message Integrity checks.

The basic public key scheme assumes that both parties have a private/public key pair, and that they can trade these pairs. Then, they use their private keys to encrypt messages to be sent, and the public ones to decrypt received messages. Additionally, message integrity can be verified when checksum is generated for each packet, signed with the private key, and sent along with the packet.

However, this kind of encryption is too time-intensive. A faster encryption scheme consists of encrypting the main part of the packet, and then sending the key used to encrypt the packet. This key is itself encrypted using the private key and so is well-protected. This way, the heavy-duty encryption is used only for the 'session key', which is very short, making the whole transaction close in speed to a non-encrypted one.

SSL adopts this latter scheme. A 'master key' is generated using some random data. The master key is used to generate a session key for each session, from which a client write key and a server write key are generated (you read with the other party's write key). The server's public key is used to encrypt the master key during the initial handshake; from then on, the packets are encrypted with the server or client write key, depending on who is sending it, a digest is taken, and the whole thing is finally packaged up with a session number.

SSL comes in two strengths, 40-bit and 128-bit, which refer to the length of the "session key" generated by every encrypted transaction. The longer the key, the more difficult it is to break the encryption code. Most browsers support 40-bit SSL sessions, and the latest browsers, including Netscape Communicator 4.0, enable users to encrypt transactions in 128-bit sessions - trillions of times stronger than 40-bit sessions. Global companies that require international transactions over the web can use global server certificates program to offer strong encryption to their customers.

SSL use in practice.

In order to use SSL between an individual using a web browser and a business hosting a website on a secured server, the following must occur:

A Root Certificate must be installed on the individual's local web browser. Certificate Authorities (CA) such as ssl.com, provide Root Certificates for public downloading by individuals.

A business must have a Server Certificate installed on their secured server. There are two ways a business can obtain access to a secured server: they can subscribe to a Secured Service Provider (SSP) such as ssl.com, or they can obtain a Server Certificate from a CA (Certification Authority). In the latter case, the business must first ask their Internet Service Provider (ISP) to generate a Certificate Signing Request (CSR), and then, the CSR must be submitted to the SSP. The SSP will verify that the business is legitimate and will issue the business a Server Certificate. This certificate is then installed on the secured server, and SSL transactions are then enabled.

Digital certificates encrypt data using Secure Sockets Layer (SSL) technology. Because SSL is built into all major browsers and web servers, simply installing a digital certificate turns on their SSL capabilities.

For further reading: the SSL site (www.ssl.com) has comprehensive documentation on the mechanics of SSL and certificates.

8.4.4 Authentication and PKI

In e-commerce/ebusiness, you need to authenticate the consumers who buy your products or services, employees who access internal systems from remote locations via the public Internet, or business partners who are tightly integrated into your supply chain and ERP systems.

For authentication, many applications choose to make use of one-time passwords. However, the use of such one-time passwords often requires the deployment of token cards -- an expensive and labor intensive effort. This is why software-based solutions are more popular. Most systems enforce authentication by developing a ***session key*** that establishes the identity of partners at session start and is used throughout a session. But then this session key needs to be encrypted. Should a private or public key system be used? Given the advantages and disadvantages of these approaches (private key is efficient but not very secure and public keys are not efficient but secure), in practice, a public key system is used to exchange the session key between the two sides. Then this key is used in a private key system only for that session. Many current systems, such as SSL (Secure Socket Layer) uses this technique. See the sidebar "SSL" for more details.

Many applications use cryptographic software to incorporate public-key cryptography for encryption and authentication. A number of such software packages exists, including the following:

- Kerberos (http://ww.mit.edu/kerberos/), a cryptographic authentication scheme using a third-party authentication server to grant cryptographic "tokens" that authenticate users to a given service. Kerberos is an open standard designed to provide strong authentication by using secret-key cryptography. Used primarily for secure interoperation of existing systems, Kerberos is used for user authentication.
- Entrust (www.entrust.com), Entrust.net, a subsidiary of Entrust Technologies that offers a portfolio of service solutions to securely manage e-business transactions. Solutions include secure e-business transactions from e-commerce Web sites to interactive cell phones. entrust also recently entered the secure transaction business for wireless. Entrust.net manages personal, web and WAP (for wireless) certificates. In particular, the new WAP Server Certificates are digital certificates that enable WAP servers to establish Wireless Transport Layer Security (WTLS) sessions with mobile phones and micro-browsers that support the WAP standard.
- PGP (Pretty Good Privacy), a popular program available on the Internet that uses public-key cryptography to authenticate users to each other without the use of certificates.
- A number of public-key based cryptographic infrastructure tools, such as the Microsoft and Netscape Certificate Servers, which allow for the inclusion of public-key certificates in various applications.

It is important to understand the role of **Public Key Infrastructure (PKI)** in authentication**.** Authentication mechanisms include a wide range of options such as user ID and password, one-time passtokens, digital certificates, and biometrics. These mechanisms are typically part of Public Key Infrastructure (PKI). PKI capabilities help create and manage asymmetric cryptographic keys or public/private key pairs required by applications. The following major PKI components provide the necessary capabilities to establish, maintain, and protect trusted relationships.

The Certification Authority (CA) creates and signs digital certificates, maintains a list of certificates that have been revoked before the expiration date (certificate revocation lists), makes these certificates and revocation lists available, and provides an interface so administrators can manage certificates.

The Registration Authority (RA) evaluates the credentials and relevant evidence that a person requesting a certificate is who they claim to be. The RA approves the request for issuance of a certificate by a CA. CA and RA functions are provided by a wide range of PKI providers such as Tivoli SecureWay Public Key Infrastructure

Directory Services, based on the Lightweight Directory Access Protocol (LDAP), define and implement a common schema for users and groups. The directory service is the point of integration for user authentication among products in many security systems. This has a positive effect on reducing administrative costs and complexity. A user can be defined once within an enterprise, and information about that user can be accessed in a consistent manner by multiple different applications. By comparison, in today's environment, common objects must be defined and administered on a per-application basis.

8.4.5 Authorization and Access Control

Authorization is concerned with assuring that only authorized users can access a particular system privilege. Authorization relies heavily on access control -- the process checking whether an authenticated user’s privileges permit the execution of a particular operation on a particular protected resource. For example, can Alice withdraw money from account zc-11-35. The access control is typically enforced through access control lists (ACLs) that may look something like the following:

User name Resource Name

Joe	Payroll
Sam	Accounting
Tim	Inventory control

Scalability of ACLs is a major issue because modern applications may scale to dozens or hundreds of Web servers and potentially tens of millions of end users. The administration of ACLs can be very complex if they must be configured on each Web server system. Authorization to back-end data or subsystems must be handled as well, including systems that have existing authorization mechanisms. In addition, authorization to other key e-business resources such as objects and message queues must be incorporated.

Due to the complexity of managing ACLs, many applications provide access control on their own because it is not always possible to provide intra-application access control using Kerberos or public-key schemes. Some products have been released that make use of the Distributed Computing Environment (DCE) access control policies. These products, such as HP's Praesidium, make use of the fine-grained access control capabilities of DCE and link them to the deployment of Kerberos within a system. Other products such as the Tivoli Secureway Policy Director provides a centralized authorization service that is the point for administering access controls for Web servers, Web applications servers, firewalls, EJBs, and other systems.

8.4.6 Accountability and Assurance

A system needs to log all attempts to access corporate resources to ensure that the system is secure. This logging can also facilitate management decisions by allowing analysis of use patterns. A comprehensive, distributed logging and audit facility for Internet-based applications is needed.

In essence, an e-business must provide assurance that the infrastructure and application resources, including systems, networks, and data, are protected with regard to confidentiality and integrity. This includes protecting the enterprise network and systems from various forms of attacks, and also requires that the communications between the consumer or business partner and the application is secure and confidential. A solution architect can choose from the set of mechanisms discussed so far to satisfy the specific security requirements for the solution. Two additional considerations are:

- Intrusion detection - These services emphasize early detection of intrusions. Should a DMZ, extranet, or any internal system be compromised, you need to detect that fact early, and take necessary actions to prevent the launching of a further attack into the private network.
- Virus detection - Computer viruses can enter your systems in a variety of ways: via e-mail attachments, from software installs, from files brought by employees from home, etc. They can quickly proliferate from system to system, user to user and cause damage to data, applications and networks. Viruses must be identified quickly, isolated, and damage repaired.

8.4.7 A Security Example

Let us go through a simple customer scenario to illustrate elements of a successful secure e-business solution.

A fictitious investment firm, Get Rich Quick (GRQ), wants to allow its customers to access and update their account information and use some of the firm's financial analysis tools via the Internet. The goal of this project is to reduce the cost of customer service. While there are many design areas at

play in this scenario, we will focus on security and how it impacts the design, deployment and management of the solution. The main restriction is that the cost of the proposed solution must be less than the projected savings in customer service.

GRQ has identified several main risks and defined mechanisms to address them. Here is a summary of three:

Risk1:

- Risk statement - Information flow, including passwords and account data, over the Internet is not secure and may be stolen.
- Mechanisms to address the risk - Ensure there are secure communications between the end user and GRQ. Make sure that all network traffic between GRQ and their customers is protected using SSL at a minimum.

Risk2:

- Risk statement- Hackers may attempt to access the GRQ system by trying user ID and password combinations to impersonate an existing customer.
- Mechanisms to address the risk - Ensure that the system can determine that users are who they say they are. Implement strong mutual authentication using PKI. Provide smart cards with certificates and smart card readers to all customers who sign up for the service and encourage their use. Provide X.509 certificates on a browser as an alternative.

Risk3:

- Risk statement- Hackers may try to attack and penetrate the GRQ network, and infect the system with a computer virus, etc.
- Mechanisms to address the risk - Protect the enterprise network and, where possible, the customer end user system from intrusion and attack. Provide antivirus software for end users who sign up for the service. Install antivirus and intrusion detection software in the enterprise. Implement a DMZ between the company intranet and the public Internet.

The application architecture uses a logical 3-tier Web application model with a thin HTML-based client that uses Java and Enterprise Java Beans (EJBs) for access to the existing customer account database. Based on this overall architecture, the following security-related design decisions were made:

- All information about users and groups is stored in a centralized directory service, deployed in the intranet, to decrease complexity and make the application easier to administer when users are added or deleted.
- A centralized authorization service is used to make it easier to define and manage the permission policy for access to programs, data, and other resources. It is deployed in the intranet. A trust relationship among the systems used by the application is used.
- The end user is required to sign on (log in) to the system once and only once. All system interaction is transparent to the end user. Credential mapping is used, where necessary, to implement single sign on.
- The system is designed to fit into the DMZ model. The application's presentation logic is deployed within the DMZ and the application's business logic is deployed within the intranet.
- GRQ will not issue certificates. A 3rd party Certificate Authority that implements Public Key Infrastructure or comparable software is used for this function.
- GRQ will log and examine attempted security breeches

Next step is to define and configure the principals and objects for each system, and their permissions, in a consistent fashion. A Security Manager (SM) is used in this step to a) define the set of Web pages and objects that will be managed, b) enroll end users / groups and server principals, c) define the permission policy for Web pages and objects, and d) add the credential mappings required for single

signon support to existing systems. Many SMs are commercially available. IBM's Security Policy Director is an example. The firewall systems are configured on each side of the DMZ. The outer firewall (router) allows only HTTP/HTTPS protocol flows, and the inner firewall allows only IIOP, and LDAP protocol flows.

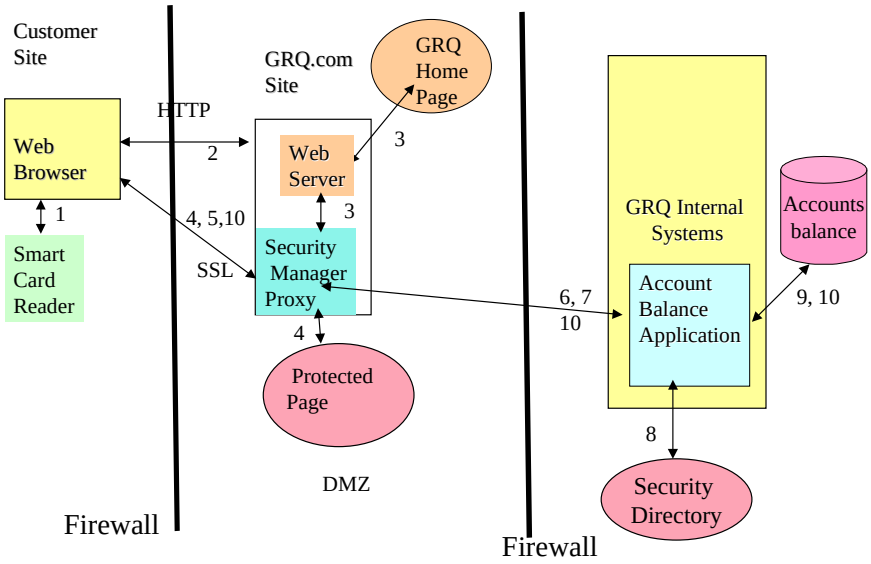


Figure 8-10: Security Flow Example

Figure 8-10 shows how the security architecture, when combined with the application architecture, results in a trusted e-business solution. The end-to-end flow sequence is as follows:

1. Pat is a GRQ customer. She inserts her Smart Card in the Smart Card Reader attached to her PC and enters her PIN number to enable her system. Pat then dials into her ISP for connection to the Internet and starts her Web browser.

2. Pat accesses the GRQ home page by typing http://www.grq.com. The HTTP request flows through the GRQ outer firewall / router to the Security Manager (SM) proxy.

3. The SM proxy inside the DMZ receives the HTTP request and determines that the GRQ home page is not protected, so the Web page is sent to Pat.

4. This home page includes a link to a protected page. By linking to this page, an SSL session is established between the browser and the SM proxy. As part of SSL processing, and to identify Pat to the SM proxy, the browser accesses Pat's certificate and private key from the smart card, which was activated in Step 1. (Note that in addition to certificates, user IDs and passwords and other third-party authentication mechanisms are also supported in typical SMs.)

5. The SM proxy sends Pat's certificate to the SM, to establish Pat's logon. The SM proxy then uses its copy of the SM access control list (ACL) to determine whether Pat has the permissions needed to access the protected Web page that lists customer applications.

6. The "Welcome" Web page is sent to Pat. It contains links to available applications. Pat clicks on the "Account Balance" application link which sends an HTTPS request to GRQ.

7. The SM proxy ensures that Pat is authorized to obtain her account balance. Once authorized, Pat's user credentials to GRQ-Accounts-Application are obtained and the HTTPS request is

forwarded to GRQ-Accounts-Application, including those credentials. This credential mapping capability provides single sign-on.

8. GRQ-Accounts-Application issues a call to the Security Directory running behind the inner firewall to authenticate Pat. This request flows through the inner firewall. This establishes Pat's logon to GRQ-Accounts-Application.

9. GRQ-Accounts-Application evaluates if Pat is authorized to execute the needed method. Several additional checks are made to make sure that Pat is authorized to access this information. Finally, an SQL query is issued and run after the database manager authenticates the credentials and authorizes access.

10. The results of the query are returned. The data is passed back through the systems to GRQ-Accounts-Application running in the DMZ where the data is formatted into a Web page which is sent to Pat over the SSL session.

Security For a Retail Sector Supply Chain - An Example

Here is an example of security considerations for a retail sector supply chain.

Corporate Security. Data access profiling according to registered company structure is provided. Specific users are identified as designated responsible participants. This ensures legitimate transactions and prevents unauthorized users.

User Security. Each individual participant is restricted to their own data. Depending on the type of transaction, users can choose either public or private access.

Transactional Security. Transactions require approval from both parties to be completed. This ensures that erroneous transactions are not completed.

Network Security. Browser connections to Web servers and apps use HTTP and secure HTTP for executing business transactions. The standard environment authenticates and encrypts communications with 128-Kbit Secure Sockets Layer (SSL) digital certificates.

8.4.8 Summary of Security

A wide range of security technologies ranging from public/private key encryptions to digital certificates and ACLs are currently available to address the authentication, protection, authorization, and accountability aspects of security. Table 8-2 shows a mapping of various security technologies to security needs (e.g., which technologies address which needs). We will discuss this table in detail in a later chapter..

Table 8-2: Security Considerations - Mapping Technologies to Needs

Technologies	Protection, Privacy, and integrity	Authentication	Authorization/access control	Accountability (Non-repudiation)
Encryption	X	X		

Password protection	X	X		
Digital signatures		X		
Message Digest	X	X		
Digital certificates	X	X		
Firewalls	X			
SSL	X			
IPSec	X			
Kerberos		X		
ACLs			X	
Audit trails				X

Time to Take a Break

- ✓ Overview & eCommerce Middleware
- ✓ eCommerce Security
- eCommmerce Platforms
- Distributed Transaction Management

Suggested Review Questions Before Proceeding

- What are the important security issues in e-commerce?
- What type of security services and protocols are needed to support e-commerce security?
- What is PKI and why is it needed for e-commerce?
- What are commerce servers (explain through examples)?
- If you had to design a new security server, what would it look like (architecturally) ?

.

8.5 Electronic Commerce Platforms: Packaging EC Middleware

8.5.1 Conceptual View of EC Platforms

Ecommerce, even just for C2B, requires a wide range of IT infrastructure services that include network services, general purpose middleware services such as Web, and EC specific services such as e-payment and EDI. It would be ideal if all the IT infrastructure services could be packaged together to support EC. This a commercial reality at present -- the IT infrastructure services needed for EC are being packaged together as **electronic commerce platforms** (also known as commerce servers). Many vendors, as we will see shortly, such as IBM, Microsoft, Oracle, Netscape, and Sun are providing such integrated EC platforms that can provide almost all services (both transactional and non-transactional) needed for EC. At a conceptual level, the EC platforms consist of the following building blocks:

- Network and operating system services
- Core middleware (e.g., web)
- EC specific middleware
- Management and support services

Electronic commerce over the Internet is relatively new at present and will evolve through various stages. A possible scenario of evolution is presented in Table 8-3. In the earlier stages, enterprises utilize services such as email and EDI. As more business processes are automated, and more business information is generated and processed in electronic form, additional services such as decision support and workflow between organizations is needed. Each stage will introduce new challenges in the infrastructure (i.e., middleware, networks, management and support services). Many business communities are currently in stage 1 of EC, although some have started experimenting with stage 2 applications. Most of the growth will be in the middleware, especially in transaction management. It can be seen that most of the growth will be in the middleware, especially on transaction and flow management between organizations. [The table shows 3 levels; here you refer to stages. Probably need to pick one term & stay with that.

The terms "Commerce Server" and "Application Server" are used interchangeably by some e-commerce platform providers. Keep in mind that an ***application server*** (also known as app server) is a platform for development, deployment, and management/support of general Web-based applications. A Commerce Server is an Application Server that specializes in e-commerce applications, thus it is a special case of the general class of application servers.

Table 8-3: Types of Commerce Servers (EC Platforms)

	Typical Examples	**Main Applications and Services**	**Network Services needed**	**Middleware Needed**	**Management and Support Services Needed**
Simple Ecommerce (Level 1)	Simple business to business commerce	• Mainly POs and invoices • Limited advertising and browsing	Internet + VAN	Email, EDI, Enhanced Fax, File transfer	Security

	Typical Examples	Main Applications and Services	Network Services needed	Middleware Needed	Management and Support Services Needed
Professional Ecommerce (Level 2)	Large electronic shopping malls, Military procurement systems	• Increased advertising and browsing • Integration within organizations	Extranet	Integration of workflows, enhanced EDI, email, legacy system integration, distributed objects	Directory Fault management
Advanced Ecommerce (Level 3)	Electronic Commerce "Brokers"	• Seamless browsing and advertising • Integration and flow of work across organizations	Extranet	Distributed Transaction Processing, Decision support, Mobile computing, Intelligent Agents, Real time multimedia	QoS

8.5.2 Conceptual Architecture of eCommerce Platforms

Figure 8-11 shows a conceptual view of a generalized ecommerce platform. At the core of this architecture is the ecommerce server that provides the set of APIs, command support, macros, daemons for dynamic web page generation, and multi-hosting for accommodating multiple virtual stores on the same machine. The ecommerce server is typically based on a web server such as Apache or IIS. Another important component of ecommerce platforms is a commerce database that houses the transactional (e.g., order information, shipping information) as well as non-transactional (e.g., tax calculation tables) catalogs and databases. These databases typically come with native support for vendor supplied databases (e.g., IBM DB2, Oracle Database, or MySQL), and can be optionally stored in any other ODBC compliant databases. Other components, also built into general purpose commerce servers, are:

- The Internet connection secure server establishes secure sessions across the Internet between the end user's browser and the ecommerce applications.
- The commerce administrator provides a suite of tools that are used to create and administer commerce sites and virtual storefronts.
- The commerce utilities support import of data into catalogs from external sources such as supplic (supplier ?) stores, and legacy data interfaces for access to business applications ("Merchant Legacy Systems") such as inventory and billing.
- The commerce companion products support related products such as Secure Electronic Transaction (SET) payments, catalog workbench for creating and maintaining catalogs, data mining tools, and document flow systems.

The conceptual view of ecommerce servers presented in Figure 8-11 can be customized and mapped to many commercially available ecommerce platforms available from IBM, Microsoft, Oracle and others. Some general trends are worth noting. For example, SOA (Service Oriented Architectures)

and WS (Web Services) are dominant. In addition, many ecommerce are becoming available as open source.

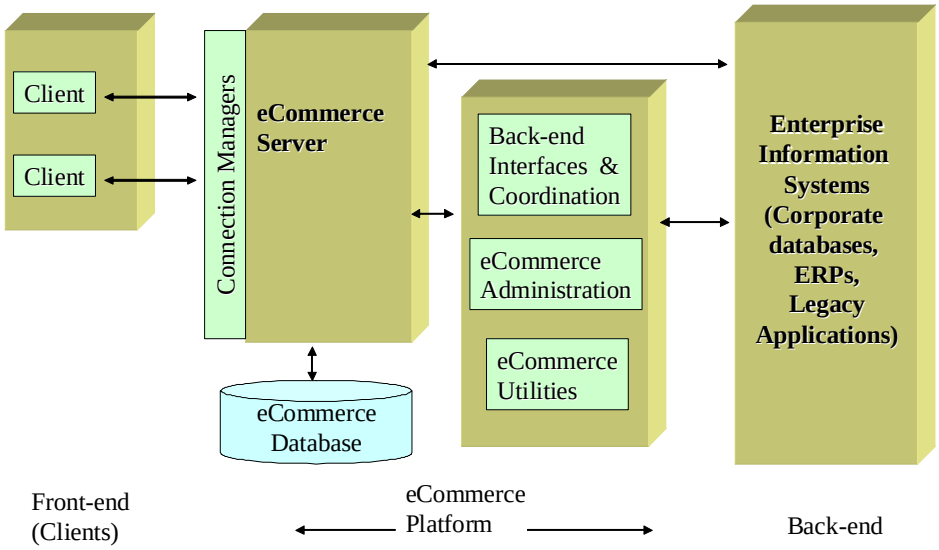


Figure 8-11. Architecture of a Generalized Commerce Server

8.5.3 Examples of eCommerce Platforms

Many ecommerce platforms have appeared in the marketplace since the late 1990s. Here is a list of the most common ones (please do a Google search to find the exact links plus more platforms):

- Oracle E-Commerce Platform
- Microsoft Internet Commerce Strategy
- IBM WebSphere
- Netscape EC Servers and the Sun iPlanet Platform
- Open Market Commerce Server
- BroadVision Commerce Platform
- Miva Merchant Server

While different vendors choose different technologies to implement their commerce servers, the following trends are worth noting:

- Most commerce servers are using Web Services as the key middleware technology
- SOA-enablement is a common strategy used in most commerce servers.
- Many commerce servers are Java based, but .Net based commerce servers also abound
- Many commerce servers are becoming available from the open source community

8.6 Case Studies and Examples

8.6.1 eCommerce for Small Businesses

Graduate students at the University of Minnesota conducted a study to learn about how small businesses use the Internet in their business (course title: "Topics in Design, Housing and Apparel: Global E-commerce", Fall Semester 2003). The students conducted 30 minute telephone interviews of Minnesota entrepreneurs to collect this information. The companies interviewed are listed below (the results of each interview can be reviewed by clicking on the links):

Andrej's European Pastry
EdVisions Cooperative
Helios Nutrition
Hunt Utilities Group
Madonna Peltier-Yawakie
Midwest Wireless
Philip Drown Companies
ProTainer
RBJ's Restaurant
Rolco Inc
Rural America Partnership
Sawbill Outfitters
Todd County Emall
Tri County Hospital
Wild Rose Farm Organics

8.6.2 International Racehorse Transport Uses eCommerce

International Racehorse Transport (IRT) transports more than 5000 horses each year around the world. Established in 1972 and headquartered in Melbourne, IRT has offices in Chicago, Auckland and LA.

The biggest challenge facing IRT was document management and profiling of each horse being transported. In fact, transportation of each horse internationally is a transaction that has to satisfy multiple customs regulations. A great deal of duplicate work and time consuming administration required employees to work long hours to keep up with the administration required by customs authorities around the world. To decrease the labor costs and time taken for each job, IRT developed a centralized database application which centrally held all information about each horse. This information is stored on servers based in the Melbourne offices and is accessible via intranet by international offices. The application is a centrally housed Lotus Notes database that can be accessed by all employees around the world, giving them access to important information required on horses being shipped to their market. The data management for each horse transported is shown below.

- The horse details are entered into an interface at the origin when the horse is registered for transportation.
- The data is stored in a database on a central server in Melbourne.
- The transport market to where the horse is being sent can review details and change what is relevant to that market.
- All parties can login and obtain the required forms to meet customs requirements.

- Needed transactions for different customs offices can be executed when a horse is transported over international borders.

The overall objective of this document management system is to increase staff morale and improve overall job satisfaction. IRT estimates 40% overall savings due to reduced administration work required for each horse. The expected savings are around $750,000 per annum due to the internal document process improvement. The development of the software application cost IRT about $200,000, the hardware cost about $75,000 and the telecommunications costs amounted to $20,000. IRT is also exploring opportunities to integrate an online booking application for clients to further save operational costs.

Reference: www.mmv.vic.gov.au/Assets/231/1/InternationalRacehorseTransport.pdf

8.6.3 HD Chauffeur Rides Uses eCommerce

HD Chauffeur Rides, an Australian company, conducts fun rides around Victoria on Harley Davidson motorcycles. HD Chauffeur Rides is a sole proprietor business with 35 experienced contract chauffeurs who conduct the rides. John Karmouche operates the business from Blackburn in Victoria, Australia.

Using photography, text and video catering for both narrowband and broadband, a website was created to demonstrate the HD Chauffeur Rides services. Booking is done online, by email or through the phone. In addition to yellow pages directory listings, brochures and business stationary, HD Chauffeur Rides' business is generated through the website's ability to be found by major search engines such as Google and Yahoo. The website demonstrates the available tours, motorcycles and quality of service by using a photo gallery as well as a streaming video of a motorcycle tour around the city. The rides can vary immensely in duration and distance and are priced after the customer requirement is understood through an interactive session.

HD Chauffeur Rides generated approximately $65,000 in sales as a result of visitors using the website. After the cost of goods used at $16,000 the net sales were around $48,750. The business was able to save $28,700 in operating expenses including two major expenses of rent, approx. $15,000 and wages, approx $13,000. The overall return on investment was around $65,290. A content management system will be implemented to automate the offers on the website. HD Chauffeur Rides is also planning to get into online marketing using search engines, pay-per-click advertising etc. They plan to increase bookings using the pay-per-click advertising method.

Reference: www.mmv.vic.gov.au/Assets/230/1/HDChauffeurRides.pdf

8.6.4 Wholesale Order & Reporting System

This case study is about a company that operates a large Ice Cream franchise business in the Southern New England and the New York City metropolitan area. Prior to a web-based wholesale ordering application all orders were handled via the phone or fax and generally the ordering process required 6 employees. Due to the efficiency introduced by the web-based application the company was able to reassign 4 employees to other functions. Application specifications included:

- Provide 100+ franchise locations with the capability to reorder supplies/products online 24/7.
- Produce on-demand (Daily, Monthly, Quarterly and Annual) reports.
- Each franchise had the potential to own several locations and each individual location needed the ability to order separately, yet all orders had to be as a sub order for the Master Store Franchise.

- Provide the ability for the each location to track their orders and to notify each location when their order has shipped via an automatic email.
- Ability to report all orders separated out first by the Master Franchise account and then by each location for that Master Franchise.
- Ability to send out automatically, monthly invoices for all orders shipped.

The application was developed by Web Global Net and used the following technologies: PHP, MySQL, Javascripting/AJAX, SSL Certificate, CSS, HTML, On-Line HTML Editor, On-Line Image Editor, Windows Server, and User Interface / Graphic Design.

8.6.5 Diary Phone – a Web-based Application to Record People's Thoughts

The Web-based audio journal application, called Diary Phone, was developed by Stylus to simplify Ecommerce. It is a 3-tier architecture-based online audio journal application that provides a cost-effective solution for capturing and recording the significant moments in people lives. Diary Phone is a web-based application that allows users to record their thoughts on the phone and delivers the audio file and transcription for a nominal fee. Diary Phone is primarily intended for the busy people who do not have the time or patience to write their thoughts down. These thoughts may include ideas about what needs to be bought from where.

When the user dials the toll free number to begin recording, he is automatically connected to the Interactive Voice Response (IVR) technology provider. Subsequently, the IVR provider is linked with the Diary Phone web application. The web application allows users to record their voice on a phone and delivers the audio file with the transcribed material to the end user. Advanced features of the website include picture gallery and the mixing of pictures and the audio to create a rich user experience.

Source: http://www.stylusinc.com/Common/SuccessStories/DairyPhone.php

8.7 Case Study: On-line Purchasing for XYZCorp

As stated previously, XYZCorp wants to setup an on-line purchasing system that will allow customers to purchase the company products through the Web. You have been asked to workout the details to make it happen. Specifically, you have been asked to do the following:

- Give an overall architecture of the system.
- Give more details about how the payment system will work through a credit card (i.e., show how a merchant bank and a credit card processing agency will participate in this system).
- Discuss how the PO processing will take place. Show the key players, their role and the flow between the players.
- How can you include security in this system? What will be the different levels of security? Discuss flow of security in this system.
- How can XML be used in this system? Give details.
- Survey, evaluate and pick an e-commerce platform that can support this storefront.
- Get the demo copy of the selected e-commerce platform (availability of a demo copy may be a selection criteria), download it and run some simple experiments to see how these platforms work.
- What will you need to do to convert this storefront into a virtual storefront (i.e., the customer can choose items from multiple suppliers)?

Hints:

Most of these questions can be answered by reviewing the material in this chapter. It is a good idea to start with the simple on-line purchasing system described in Section 8.2.3. You should draw a sketch that resembles the following figure and then translate it into a physical architecture ("solution architecture") that shows the middleware, and the network.

It is a good idea to develop an evaluation criteria before conducting a survey of e-commerce platforms. The evaluation criteria should include factors such as features provided, extensibility, interfaces with other systems, availability of a demo copy, number of users, vendor support and staying power.

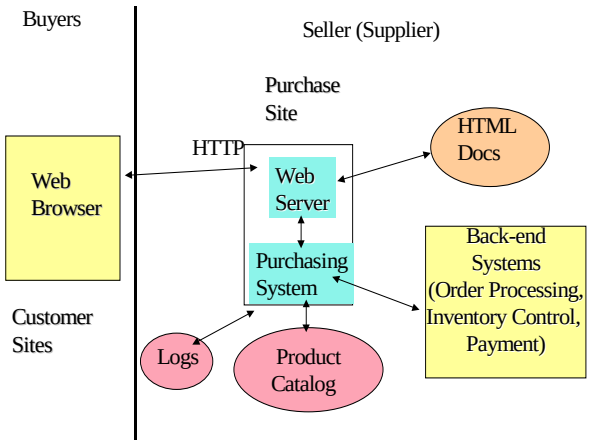


Figure 8-12: A Simple Internet-based Purchasing System

8.8 Concluding Comments

This chapter has given an overview of the middleware services and platforms needed to support EC. At a conceptual level, the IT infrastructure for EC consists of the following :

- **Networking services** to provide the network transport between EC partners.
- **General purpose Middleware services (**e.g., web services**)** to support interactions between remotely located, including but not restricted to, EC partners. These services are not discussed in this chapter (we discussed them in the Chapter "Web and XML").
- **Basic EC middleware services** to provide value added features needed by many EC applications. Examples of these services are purchasing, payment, shopping carts, billing, EDI and XML. These services have been discussed in this chapter.
- **Advanced EC/EB middleware services** needed to support the virtual/next generation enterprises. Examples of these services include: support for mobility, supply chain management, enterprise integration software, and electronic markets/trading hubs. These services are discussed in other chapters.
- **Management and support services** for administering the electronic enterprises. Examples of these services include planning, provisioning, fault management, performance management, change management, and security for electronic enterprises.

E-commerce is often a high-risk business. One way to mitigate high risk is by limiting assets, and this is accomplished by outsourcing the infrastructure. At present, almost anything can be outsourced, from a simple merchant account to an entire Web E-commerce site. Outsourcing, it should be noted, shows its negative side by impacting return margins. So, while outsourcing the infrastructure is a viable alternative during the initial phase of an e-commerce venture, it may not be adequate during the growth and mature phases, where a more efficient process would guarantee higher return margins.

8.9 Review Questions and Exercises

1) Find and describe an on-line purchasing system being used in the industry.
2) What are the key attributes of EC middleware? Which ones are not discussed in this chapter?
3) Discuss the role of transaction processing in e-commerce.
4) What security packages are commercially available to satisfy EC security requirements?
5) Choose two commerce servers discussed in this chapter and compare/contrast.
6) Choose two commercially available commerce servers and compare contrast them.
7) Conduct a literature survey and identify additional commerce servers not discussed in this chapter.

Time to Take a Break

- ✓ Overview & eCommerce Middleware
- ✓ eCommerce Security
- ✓ eCommmerce Platforms
- Distributed Transaction Management

8.10 Attachment A: Distributed Transaction Management Details

8.10.1 Overview of Transaction Management Concepts

Let us now turn our attention to data modification that is typically accomplished through a transaction manager. The concept of a transaction originates from the field of contract law [Gray 1981, Walpole 1987] in which each contract between two parties (a transaction) is carried out unless either party is willing to break the law. From a business and end-user point of view, transactions occur at two levels: customer to business and business to business (see Figure 8-13). In computer science, a transaction is defined as a sequence of data operations (read, write and manipulation commands) that transform one consistent state of the system into a new consistent state [Eswaran 1976]. Examples of business transactions are: electronic transfer of money from one account to another, update of an inventory database, and purchasing a ticket electronically. To accomplish these business transactions, computer transactions are executed. Examples of the computer transactions are a group of database operations (e.g., SQL statements) that need to be executed as a single unit, a program with embedded SQL statements that updates one or more relational tables, and a Cobol program that modifies indexed files [Ozsu 1999].

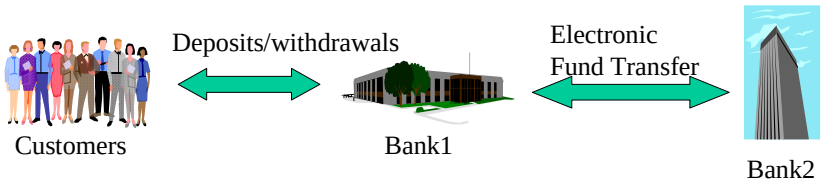


Figure 8-13: Transaction Example: Customer to Business and Business to Business

8.10.1.1 The ACID Properties

As discussed previously, a transaction has four properties, known as the ACID (atomicity, consistency, isolation, durability) properties. Detailed discussion of the ACID properties for transactions can be found in [Gray 1993, Ozsu 1999]. The implication of the ACID properties for transaction management is as follows:

Serializability (Concurrency Control): This allows transactions to execute concurrently while achieving the same logical result as if they had executed serially. Concurrency control allows multiple transactions to read and update data simultaneously, and includes transaction scheduling and management of the resources needed by transactions during execution. Transactions can be scheduled serially ("single-threaded") to minimize conflicts or in parallel ("multi-threaded") to maximize concurrency.

Commit Processing: This allows commitment of transaction changes if it executes properly and removal of the changes if the transaction fails. The transactions usually "bracket" their operations by using "begin transaction" and "end transaction" statements. The transaction manager permanently enters the changes made by a transaction when it encounters the "end transaction" statement; otherwise, it removes the changes. Transaction managers also log the results of transactions on a separate medium so that the effects of transactions can be recovered even in the event of a crash that destroys the database.

Although the ACID properties as well as the serializability and commit processing implications are important, it has been argued that all these properties amount to atomicity and serializability [Triantafillou 1995]. This is a pragmatic view that greatly simplifies the discussion of transaction processing.

8.10.1.2 Transaction Models

Transactions may be classified into the following broad models:

- **Single-site versus multiple-site (distributed) transactions**: The transactions may be restricted to a single site (e.g., one database server) or it may span many sites. We will discuss distributed transactions in the next section.
- **Queued or conversational transactions**: In queued transaction processing, such as found in IMS-DC, arriving transactions are first queued and then scheduled for execution. Once execution begins, the transaction does not interact with the user. In conversational transaction processing, such as found in CICS and IMS, the transactions interact with the outside world during execution.
- **Short (flat) or long (workflow) transactions**: Short, also known as flat, transactions start with a "begin transaction" instruction and end with a "commit transaction" or "abort transaction" instruction. Flat transactions are all or nothing at all activities (you cannot commit a portion of a flat transaction). Long duration transactions (also known as Workflows, Sagas and Flexible Transactions) consist of a sequence of distributed or queued transactions to perform a multitude of activities that may span several business units of an organization. Long transactions may be constructed by chaining or nesting individual transactions. These transactions cannot be typically

satisfied by a single transaction (distributed or queued). A special case of long transactions is massive batch updates. These transactions are typically handled by providing a series of "synch points" at which all the changes made are committed (e.g., a synch point after every 100 hundred updates).

8.10.1.3 Transaction Managers

A transaction manager (TM), also known as a transaction processing monitor (TP monitor), specializes in managing transactions from their point of origin to their termination (planned or unplanned). The TM facilities are traditionally integrated with the DBMS facilities, as shown in Figure 8-14. This allows database queries from different transactions to access/update one or several data items. However, some products only specialize in TM with special focus on handling thousands of OLTP users. These TMs provide a variety of monitoring, dynamic load balancing, process restarts, and priority scheduling capabilities. Under the control of a sophisticated TM, a transaction may be decomposed into sub-transactions to optimize I/O and/or response time. Examples of commercially available TMs are CICS, Encina, and TuxedoIt is not always possible to find separate TMs in commercial products. In some systems, TM facilities are embedded in communication managers, operating systems and/or database managers. In particular, most RDBMS vendors at present provide some TP facilities, known as TP-Lite (see Section 8.10.5).

An introduction to TMS facilities is given in the book [Ozsu 1999]. A detailed classification of transaction processing systems can be found in [Leff 1991].

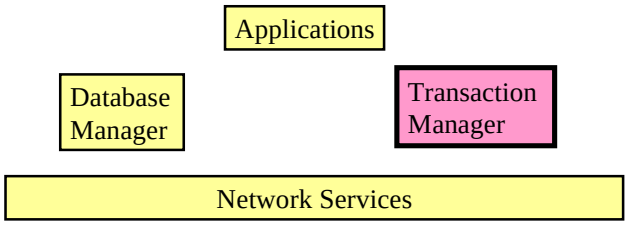


Figure 8-14: Conceptual View of a Transaction Manager

8.10.2 Distributed Transaction Processing Concepts

Distributed transaction processing (DTP) allows multiple computers to coordinate the execution of a single transaction. This occurs when the data needed by a transaction resides at many computers. Atomicity of a transaction is of key importance - all the activities performed on different computers by a transaction must be completed properly or entirely withdrawn in the event of a failure in the network, application code, and/or computing hardware. A distributed transaction manager (DTM), a collection of software modules, is responsible for distributed transaction processing. Transactions in a DTM are known as distributed (also known as multi-site) transactions that access data at several different sites. A distributed transaction consists of several local (also known as single site) transactions which access data at one site.

8.10.2.1 Distributed ACID

Each distributed transaction is treated as a single recoverable unit and must pass the "ACID" (atomicity, consistency, isolation, and durability) test. Consequently, the main responsibilities of a DTM are as follows:

- **Atomicity** of transactions through commit processing for failure handling and recovery.
- **Serializability** of transactions through update synchronization and concurrency control.

It is important for different sites to reach commit agreement while processing sub-transactions of a global transaction. The most widely used solution to this problem is the two-phase commit (2PC) protocol that coordinates the commit actions needed to run a distributed transaction. When a transaction issues a COMMIT request, the commit action is performed in two phases: prepare for commit and then commit. If a failure occurs in the prepare phase, then the transaction can be terminated without difficulty; otherwise, all sub-transactions are undone. Please note that phase 2 is COMMIT and must complete. Two-phase commit will be discussed in Section 8.10.4.1. Two-phase commit has been implemented in many systems and is also included in the ISO Transaction Processing (TP) standard.

Many algorithms for update synchronization and concurrency control have been proposed and implemented since the mid 1970s. Most algorithms used in practice are variants of two-phase locking (2PL), which allows a transaction to lock the resources in first phase and unlock in the second phase after performing reads/writes. Algorithms are also used to resolve distributed deadlocks that occur when transactions wait on each other. A review of these algorithms can be found in [Umar 1993 Chapter 6].

8.10.2.2 Distributed Transaction Models

Figure 8-15 shows a few basic models of distributed transactions. In case of remote transactions, the client submits (ships) the request to execute the transaction on a remote system. The remote transaction either commits or aborts, independent of the requesting system. In case of commit coordination, the requesting site manages the execution of the transaction across multiple sites. The protocol used in this case is two-phase commit. Serial execution moves the coordination from one site to the next to complete a multi-site transaction.

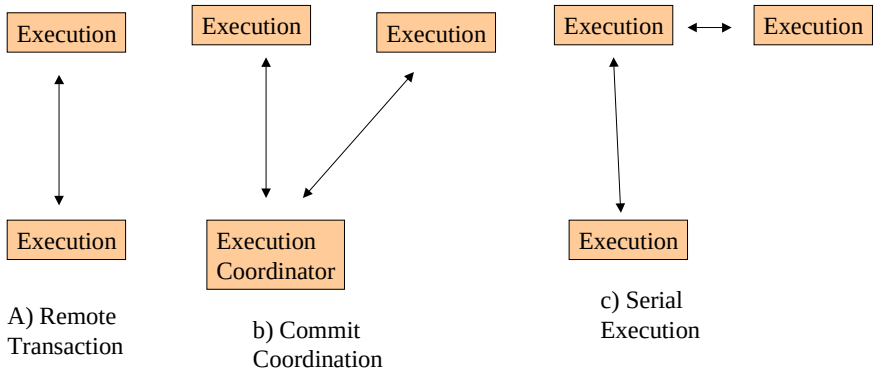


Figure 8-15: Models of Distributed Transactions

These three basic models can be combined to produce many other DTP models for long running (workflow) distributed transactions. In addition, the activities performed on different systems can be coordinated as queued or conversational transactions. In queued DTP, the transaction managers at different sites queue the incoming transactions and then execute them later, thus allowing for organizational boundaries and control between systems. In conversational DTP, the transaction managers interact with each other directly through the communication network. Note that queued DTP is not suitable for commit coordination (Figure 8-15b) because the sending site can only communicate with other sites through queues. Queued model does work quite well for remote and serial transactions.

8.10.2.3 Distributed Transaction Managers: The TP-Heavy Approach

A distributed transaction manager (DTM) is responsible for the execution of a distributed transaction from its beginning to its end and is in charge of ACID in a distributed environment. Thus, a DTM must address numerous technically challenging issues such as concurrency control in distributed environments, update synchronization, and data integrity after failures in a distributed environment. Due to the complexity of DTMs, they are referred to as "TP-Heavy". Research in distributed transaction management has been actively pursued for heterogeneous databases for several years [Pu 1991, Soparker 1991]. Although detailed technical discussion of these topics is beyond the scope of this book, we will review the key technical challenges and approaches in Section 8.10.5 and suggest additional sources of information.

Examples of DTM products are Microsoft's Transaction Processor, NCR's Top-End, BEA's Tuxedo and Transarc. It is expected that in the future more products will become available and products will operate across PCs, UNIX and MVS (newer versions are called Z/OS) environments. In addition, these products will conform to the ISO and X/Open standards for distributed transaction management. We discuss these standards next. Some interesting references in this area are:

- Vogler, H., and Buchmann, B., "Using Multiple Mobile Agents for Distributed Transactions". CoopIS 1998: 114-121.
- Weikum, G., "Review - Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce". ACM SIGMOD Digital Review 1:(1999).
- Ram, P., Do, L., and Drew, P., "Distributed Transactions in Practice".SIGMOD Record 28(3): 49-55 (1999).

Web Services Transactions

As indicated in a previous chapter, Web Services (WS) is an area of tremendous activity at present. Two specifications (WS-Coordination and WS-Transactions) are addressing the reliable, transactional coordination of Web Services. WS-Coordination defines a general framework for coordination between Web Services with a shared context (e.g., a common databases). However, WS-Transaction defines two particular coordination types for (short-running) atomic transactions and (log-running) business transactions. A third development is the Business Transaction Protocol (BTP) from OASIS that addresses long-running business transactions and deals with ACID properties accordingly. Work is proceeding on all specifications.

Sources for additional information:

- Little, M., “Transactions and Web Services”, CACM, Oct. 2003
- “Web Services Coordination Specification”, www.ibm.com/developerworks/library/ws-coor/
- “Web Services Transaction Specification”, www.ibm.com/developerworks/library/ws-transpec/

8.10.3 Data Replication Servers – The TP-Lite Approach

8.10.3.1 Overview

Data replication is concerned with copying data completely or partially to multiple sites. The main advantage of replication is that the data is stored where it is used most often. In addition, replicated data can be accessed from alternate sites. Replication improves the read performance and data availability, but can degrade the update performance due to the need to synchronize replicated copies. Depending on the business needs, data may be replicated to support operational processing or informational processing. Here are some examples of data replication:

- Price information is replicated at all stores to speed up the checkout counter processing. The price information may be completely or partially replicated.
- Skeleton customer information (e.g., customer name, account number, credit limit) is kept at all different stores to speed up the order processing. Complete customer information is kept at a central site.
- Data warehouses containing portions of data from operational systems are constructed by many enterprises to support decisions in marketing and business planning.
- Detachable computers (e.g., mobile computers) are not always connected to the data source and therefore, typically keep redundant data. Detachable computers usually extract needed data and store it on their local disk for access while operating in a detached mode.

Data may be replicated for the following reasons [White 1994, Triantafillou 1995]:

- Distribute data to where it is used to improve performance, e.g., the price file.
- Data may be replicated to improve data availability.
- Data may be copied to detachable computers for off-line processing.
- Data from different decentralized servers or detachable clients may be integrated into one copy.
- Data is extracted from multiple operational databases and loaded/replicated into a data warehouse.
- The database on some platforms may have better tools for application development and/or administration. Thus, it may be quicker and cheaper to develop and maintain new applications around the replicated data on new platforms.
- Some platforms may provide better and easier data access and manipulation tools.
- Data may be replicated during a gradual migration period.

However, data replication can be an expensive and time-consuming activity that must be carefully coordinated. In particular, the issue of keeping the replicated data synchronized with master/primary data is of key importance.

8.10.3.2 General Architecture of Replication Servers

Figure 8-16 shows a set of logical components of a generalized Data Replication Server. In essence, a data replication server provides software components that replicate the changes being made at a primary site to one or more secondary sites. These logical components are customized and specialized by different vendors depending on how the data is initially loaded, how it is synchronized (refreshed or incrementally updated), whether the replicated data is read-only or updateable, how the data changes are captured, and how frequently the data is synchronized.

Data Extractor: The data extractor component is responsible for selection (capture) of needed data from the primary databases. Entire database may be selected for refresh or only deltas (i.e., data changes) may be captured. Data captures for refresh is straightforward, but requires considerable effort for delta captures. The delta captures can be achieved through a variety of techniques such as the following:

- Capture the data changes from the logs. The capture program continuously monitors the logs, and extracts the needed changes. This technique is commonly known as "log scraping".
- Use database triggers, commonly available in RDBMSs, to capture the changes. The triggers can be set for times (e.g., 6 p.m. every day) or other events such as whenever a new order arrives.
- Employ programmatic captures from the database. A capture program scans the source databases and selects data based on predefined business rules. The capture logic is application dependent and not part of the replication server. An example of a business rule is: "extract all data since yesterday 5 p.m. on the highest sales in the southwestern region of the company".

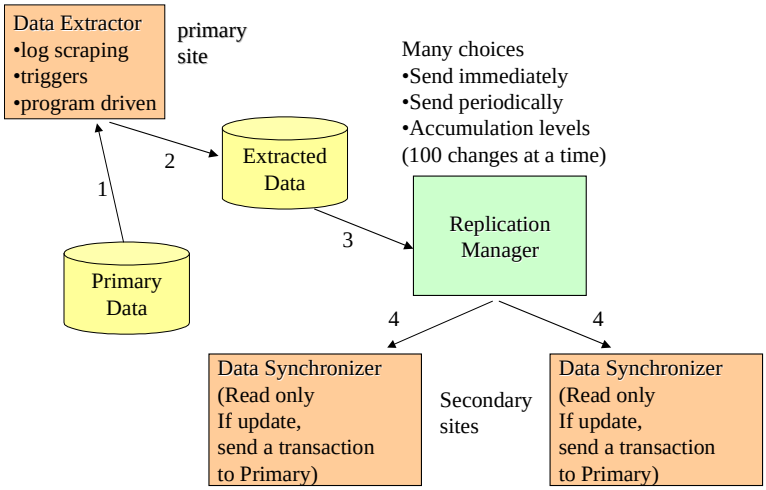


Figure 8-16: Data Replication Server Architecture

There are many trade-offs between these approaches. The log scrapers are off-line processes and do not interfere with the operation of the system. The triggers provide more flexibility but require extra programming effort. The specialized data capture programming gives maximum flexibility to the end-users, but can be an expensive undertaking.

The data extractor component may also transform the primary data into the secondary data formats, if needed. This includes conversions of data formats and data models (e.g., IMS to RDBMS). Many data replicators convert data to ASCII format and/or transform it into a target data load format. For most of the data replications, the need for extensive data conversion may not exist because the same data in the same format may be replicated at several sites (e.g., customer information at multiple sites). However, some applications such as data warehousing require considerable transformations such as data consolidation (unification of different values), summarization, and derivation (generation of new fields).

Replication Manager: This component is the heart of the data replication server. It receives the data from the primary site(s) and transmits it to the appropriate secondary site(s). The replication manager may itself reside on the primary site(s), on a separate "replication server machine", on secondary site(s), or on a combination. Associated with the replication manager is a directory which shows what data is to be extracted, where it is to be sent (i.e., the secondary site(s)), how frequently the extracted data is to be sent to the secondary site(s)), and if the sent data is to be applied immediately or delayed.

The replication manager usually includes an administrative module that enables users to define, generate, initialize, and customize the various replication server components. Examples of the administrative functionalities are:

- Generation of data extraction and conversion programs.

- Specification of the primary and secondary sites for the replicated data items.
- Specification of the parameters to govern the operation of various components.

The administrative functionality can be provided through a GUI operating from a desktop. The generated programs can run on a desktop or on the primary data site. The administrative facilities may also include scripting languages to ease the burden of operating and administering a Data Replication Server.

Data Synchronizer: This component is responsible for transmission of captured data from the primary site to the secondary site(s) and resultant data load/update at the secondary site(s). The secondary data may be replaced entirely with the new data (refreshed) or it may be updated selectively to reflect changes (deltas). In many data replication servers, a staging area is created on the primary sites to store the captured data. Data synchronizers employ remote data transfer service such as bulk data transfer or interactive data exchanges depending on how quickly the secondary data is to be synchronized. Depending on the amount of data to be transmitted and loaded, special techniques may be used to speed up the transmission/load process (e.g., bulk database load). Typical options for this component are:

- Establish client/server sessions between the primary and secondary sites to apply the updates in real time as they become available or employ bulk data transfer for batch update processing (e.g., end of day processing).
- Resolve conflicts, if applicable. Many older replicators allowed updates of primary copy only, thus no conflict resolution was needed. Newer replication servers (e.g., the latest Oracle Replicator) are beginning to add "advanced" features that allow multiple copies to be updated based on a conflict resolution scheme. Typical conflict resolution schemes are based on time stamps (i.e., if copy A is being updated but copy B has more recent changes, then A is not updated and the request is sent to copy B).
- Broadcast the data to all secondary sites simultaneously or serialize the updates one secondary site at a time.
- Lock all secondary sites when they are being updated to assure that all secondary sites have the same copy of data or ignore this locking.
- Apply updates immediately when they are received at the secondary sites or perform bulk updates periodically.
- Use bulk load utilities and/or SQL load for RDBMSs.

Framework for Analyzing/Selecting Data Replication Servers. Several data replication servers are becoming commercially available from a diverse array of vendors. Here is a partial list of data replication servers, listed alphabetically:

- IBM Data Replication Toolset (Data Refresher, Data Hub, Data Propagator)
- Oracle Replicator
- Prism Warehouse manager
- Sybase Replication server
- Trinzic Infopump and InfoHub
- Praxis Omni Replicator

Some of these servers are oriented towards data warehousing while the others are intended for distributed transaction processing. It is beyond the scope of this book to discuss these and other data replication products. However, the generic architecture presented in the previous section can be used as a basis to analyze, evaluate, and select the most appropriate data replication tools. For example, the generic architecture can be used to analyze the data replication servers based on the following factors:

- Type of primary data sources
- Type of secondary sites (read-only or update)
- Mechanism to capture data changes (e.g., log scrapers versus triggers)

- Data refresh and/or delta update capabilities
- Real-time versus periodic batch update synchronization
- Conflict resolution options

Table 8-4 shows the factors that can be used to analyze/evaluate data replication servers. These factors are presented in terms of the logical architectural components presented in the previous section. Table 8-4 can be used as a basis for generating questions that can be included in a request for proposal (RFP).

Table 8-4: Analysis Matrix For Data Replication Servers

Evaluation Factors	Product1	Product2	Product3
Extraction capabilities ▪ Primary data supported (e.g., IMS, DB2, Oracle, Sybase) ▪ Ability to select records/fields from primary databases ▪ Method of data extraction (log scraping, triggers, programs ▪ Any data conversions performed			
Server management/administration capabilities ▪ Automatic extraction/conversion of code generation capability ▪ Quality of code generated ▪ Completeness of code generated ▪ Efficiency of generated code ▪ Ease of use (user GUI interface, training requirements) ▪ Data dictionary capability (ties to data dictionaries) ▪ Productivity aids (e.g., generation of scripts, JCL created)			
Update synchronization capabilities ▪ Client/server sessions between the primary and secondary sites or bulk data transfers ▪ Broadcast the data to all secondary sites or serialize ▪ Lock all secondary tables (or rows) when they are being updated or ignore this locking ▪ Apply updates immediately when they are received at the secondary sites or perform bulk updates periodically ▪ Use bulk load utilities and/or SQL load for RDBMSs			
Operability (environment needed for tool operation, e.g., PCs, UNIX workstations)			

Vendor maturity (strength of company, market position, etc.)			
Replication server maturity (product reliability, product direction)			
Product support (based on other client feedback and customer service contracts, technical support, help desk support)			

8.10.4 Two -Phase Commit ("TP-Heavy") Versus Data Replication Servers ("TP-Lite")

Implementation of algorithms for failure handling of distributed transactions is an expensive undertaking. Two-phase commit (2PC) and Data Replication Servers are two different approaches to guarantee integrity of distributed data under failures. Two-phase commit as well as Replication Servers have some trade-offs. Due to the academic and industrial activity in two-phase commit and the widespread availability of Replication Servers, a careful analysis of the trade-offs between using two-phase commit-based algorithms versus using Replication Servers is essential for several practical situations.

The reader should keep in mind that both issues are transparent to the end users. The discussion in this section is intended for a general understanding of the issues and approaches that should lead to improved strategies for the overall architecture.

8.10.4.1 Two-Phase Commit

Two-phase commit is the principal method of ensuring atomocity of distributed transactions. For a single-site transaction, updates are made permanent when a transaction commits, and updates are rolled back if a transaction aborts. However, a distributed transaction may commit at one node and abort at another. For example, an update completes at node n1 and fails at n2. A transaction, distributed or not, may terminate abnormally due to two reasons: "suicide", indicating that a transaction terminates due to an internal error like a program error, or "murder" to indicate an external error like system crash [Gray 1979]. It is the responsibility of two-phase commit software to remove all changes made by a failing transaction from all nodes so that the transaction can be re-initiated. For atomic actions to be recoverable, the following two conditions must be met:

- Updated objects are not released until the action is completed.
- The initial states of all objects modified by the action can be reconstructed through the use of a log.

The two-phase commit protocol adheres to these conditions and coordinates the commit actions needed to run a transaction. When a transaction issues a COMMIT request, a series of actions are initiated. These actions are divided in the following two distinct phases:

Phase 1 (Prepare): This phase is preparatory, the commit is not actually carried out in this phase. The participating sites in this phase record enough information in the logs so that a transaction can be rolled back or committed, if needed. The specific steps in this phase are:

1. The "commit global coordinator" (the initiating node) sends a PREPARE message to all cohorts (commit coordinators running on nodes participating in the execution of this transaction).

2. Each cohort logs enough information so that it can roll back or commit the transaction.

3. Each cohort sends one of the following responses to the global coordinator:

"prepared" - the modified data has been prepared for commit or rollback.

"read-only" - no data on the node has been modified, so no prepare is needed.

"abort" - the node cannot successfully prepare.

4. The global coordinator waits for a reply from all cohorts.

5. If all cohorts indicate "prepared", then the next phase is initiated; if any cohort indicates "abort", then the entire transaction is rolled back at all sites; if a cohort indicates "read-only", then that cohort is bypassed from commit processing (this expedites the two-phase commit processing).

Phase 2 (Commit): If all cohorts respond "prepared" and/or "read-only", then the initiating site (global coordinator):

1. Performs Create/Update/Insert/Delete and writes COMMIT entry into the log.

2. Sends a COMMIT message to each cohort.

3. Waits for positive response from each cohort. If no response, then write abort message and terminate the transaction.

4. Writes a complete entry in log and terminates.

The key problem in two-phase commit is failure of the global coordinator (i.e., the originating node fails during commit processing). In addition, two-phase commit causes tremendous delays and "discomfort" in an unreliable environment (it is tough to succeed in two-phase commit if any cohort fails for one reason or another!).

The protocol described here shows the basic two-phase commit processing. A great deal of intricate processing takes place at the originating node and cohorts (see [Gray 1993] for details). Two-phase commit (2PC) has been implemented in several systems with some variations to deal with different failure conditions. In general, two-phase commit is offered by DBMS vendors as automatic (i.e., totally done on behalf of application developers) or programmatic (i.e., 2PC subroutines provided to application developers for customized usage). An extensive discussion of the reliability issues for no data replication, data replication, full replication, and network partitioning is given by [Garcia-Molina 1987].

8.10.4.2 Trade-offs Between Two Phase Commit and Replication Servers

Two-phase commit (2PC) and Data Replication Servers are two different approaches to guarantee integrity of distributed and replicated data under failures. The fundamental difference between these two approaches is the transaction versus periodic propagation of updates (update synchronization).

Transaction level update synchronization is the basis of 2PC. In this synchronization scheme, a distributed transaction is treated as an atomic action and thus, all updates must be synchronized during a distributed transaction or the entire transaction must be rolled back. This is the basic reason for the somewhat complicated steps of 2PC. Periodic update synchronization, on the other hand, synchronizes updates after completion of a transaction. This approach, used in the commonly available Data Replication Servers, does not respect the boundaries of distributed transactions (i.e., data is synchronized after a transaction has completed).

Both approaches have certain advantages and disadvantages. 2PC has the following major pluses and minuses:

+ All updates are simultaneously available to end-users.

+ Guarantees atomicity of a distributed transaction.

+ Fully transparent to end-users.

+ Many improvements have been introduced to increase robustness, flexibility, and efficiency.

- Chances of failures are high in unreliable systems (transactions abort too many times).

- Takes too long if many copies of data exist (too many cohorts).

- Creates difficult situations if the global commit coordinator (the originating site) fails during 2PC.

- Does not allow many customizations and complicated application rules (e.g., retry prepare if cohort responds with "abort", apply an update after certain time).

Owing to these limitations of 2PC, Replication Servers are becoming a viable alternative for many organizations. However, the periodic update synchronization scheme used in many Replication Servers also has some pluses and minuses:

+ Very flexible (can be configured for different situations such as events, time, triggers, etc.).

+ Can offer additional capabilities (i.e., conversion and transformation of data).

+ Can be used for large and occasionally unreliable networks.

- Requires a primary/secondary copy (this paradigm may not fit some applications). This restriction is being removed in modern replication servers by introducing conflict resolutions that are based on rules or manual intervention.

- Some notion of global time must be maintained to assure that updates are synchronized at certain times.

Owing to these trade-offs, some systems provide transaction as well as periodic update synchronization. For example, several Replication Servers allow transactional update synchronization that employs 2PC. It should be noted that security is a serious consideration for replication. That is, more copies of the data means more places for a hacker to visit and corrupt. In addition, update intensive applications may be better candidates for replication since 2PC may just become too unwieldy and time consuming.

The following guidelines are suggested to the users of 2PC and Replication Servers [McGovern 1993]:

- Keep data replication as minimal as possible. Large number of replicates can cause serious problems in 2PC as well as Replication Servers.
- If data must be synchronized as a transaction, then keep the number of copies small and use 2PC.
- If concurrency requirements outweigh "subsecond" data integrity requirements (i.e., data can be synchronized periodically) then use Replication Servers.
- If the network and nodes are unreliable, then use Replication Servers.

Extensive discussion of this topic can be found in [Schussel 1994, Gray 1993, Garcia-Monila 1987].

8.10.5 Distributed Transaction Processing: TP-Less, TP-Lite, TP-Heavy

In distributed environments, approaches to handle transactions (data update) vary widely due to the wide range of configurations (small PC LAN-based systems versus large systems involving multiple mainframes), query versus update traffic (ad hoc SQL queries versus massive updates), vendor offerings (database vendors versus TP vendors), and user/developer background (PC users/developers versus mainframe users/developers). The approaches being used at present fall into the following categories:

- TP-Less, i.e., do not use any transaction management considerations.
- TP- Lite, i.e., use database procedures to handle updates.
- TP-Heavy, i.e., use a transaction manager to handle updates.

Another approach, somewhere between TP-Lite and TP-Heavy (perhaps TP-medium), is becoming increasingly popular due to the growth in messaging-oriented middleware (MOM). See the sidebar "Running to MOM for Distributed Transaction Processing".

8.10.5.1 TP-Less

In this case, the database and file management capabilities for retrieving and updating data are used. For example, some relational database vendors treat each SQL statement as a transaction. Thus, each SQL select, update, insert, and delete is treated as a unit of consistency. However, a group of SQL statements are not combined into a transaction that must be committed or aborted. Similarly, in file systems, each file read and update is treated by the users as a transaction (there is a potentially serious problem here because many file systems do not provide the capabilities to treat each file I/O as a transaction).

TP-Less is currently being used heavily in small C/S environments with PCs and UNIX machines. In particular, this approach is favored heavily when all data needed by the users is on one SQL server. TP-Less has the advantage of being efficient and inexpensive (no additional overhead and software is needed). However, it has several limitations. First, it cannot be used when some data is in flat files (the ability of file managers to provide ACID properties should be examined carefully). Second, related updates cannot be grouped together as a transaction. Finally, all data must reside on one site (TP-Less may use a data replication server to handle duplicate data on multiple sites).

8.10.5.2 TP-Lite

TP-Lite goes a step beyond TP-Less by implementing each transaction as a stored procedure. Recall that a stored procedure is a collection of SQL statements that are performed as a unit (they may also have some non-SQL logic contained in them). A user can define, for example, a set of SQL statements that update a customer account and store them in a database management system as a stored procedure that is invoked from different programs that need to update customer account information. Any updates that need to be performed together and any retrievals that are dependent on these updates can be imbedded in a stored procedure. In addition, stored procedures can enforce any additional integrity and security restrictions. TP-Lite capabilities are provided by most RDBMS vendors such as Informix, Oracle, and Sybase (basically, any vendor that supports stored procedures is in the TP-Lite business).

TP-Lite is mainly the invention of database vendors to provide some transaction management capabilities. This approach, when combined with data replication servers to synchronize replicated data, appeals to many C/S application developers. TP-Lite is being widely used to manage transactions in C/S environments where the data resides on a single SQL server. TP-Lite is better than TP-Less (it supports a group of SQL statements as a transaction), but it does not provide any global

transaction control. In addition, TP-Lite cannot be used to handle transactions that need access to data stored in flat files (stored procedures are the domain of database vendors).

Running to MOM for Distributed Transaction Processing

Message-oriented middleware (MOM) is gaining popularity for many applications, including light-weight implementations of distributed transaction processing (DTP).

MOM, discussed in detail in a previous chapter, allows an application A to put a message on a queue that is later picked up by application B (or C and D) to process asynchronously. A queue can be a print stream or any intermediate file. This simple approach can be used to link existing applications very easily without modifying any code at either side (i.e., redirect the output of application A to a disk queue and redirect the input of system B to the same disk queue). This approach does not require the additional software development on either side (you do not need a client that issues a RPC and a server that receives, parses and dispatches processes). This also eliminates the need for staff training on both sides.

The main appeal of MOM for DTP is that the queue messaging can be transactional (i.e., MOM can make sure that only one message is transferred and that automated rollback recovery is available). This is a very familiar territory for mainframe-based transaction managers such as IMS (IMS has been using queued messages since the 1970s).

MOM providers such as IBM, DEC, Peer Logic and Covia Technologies are actively pursuing this opportunity.

8.10.5.3 TP-Heavy

TP-Heavy uses a separate TM to manage transactions in C/S environments. As discussed previously in this chapter, these TMs maintain the ACID properties of transactions that may span many database servers (i.e., they support DTP). For example, they allow PCs to initiate complex multi-server transactions from the desktop. TP-Heavy systems support the DTP functions discussed earlier in this chapter (i.e., global concurrency control, distributed two-phase commit, failure handling). More importantly, TP-Heavy systems are not restricted to database transactions -- they manage all data (flat files, databases, and queues). Examples of TP-Heavy products for C/S environments include CICS, Encina, Tuxedo, and Top End.

TP-Heavy has the obvious appeal since it takes transaction management seriously. TP-Heavy is essential when transactions involve data stored in multiple formats on multiple sites. However, TP-Heavy may be too "heavy" for small C/S applications that need access to data stored on a single SQL server.

8.10.5.4 Trade-offs Between TP-Lite and TP-Heavy

It appears that TP-Lite as well as TP-Heavy have certain pluses and minuses in C/S environments (TP-Less is too restricted for most serious business applications). The following questions should be asked by an application developer before deciding on TP-Lite versus TP-Heavy:

- In what format is the data stored (databases, flat files)? If the data is stored in multiple databases and flat files, then TP-Lite is not suitable (database procedures only work in RDBMS environments).

- How many SQL servers does the data reside on? If the application needs to update and commit data that is stored on multiple servers, then TP-Heavy should be used (database procedures cannot participate with other database procedures in a distributed transaction).
- What is the requirement for data synchronization? If the data synchronization interval is periodic, then a TP-Lite solution combined with a data replication server may be useful to handle updates against replicated data.
- What are the requirements for performance and load balancing? TP-Lite solutions with database procedures are much faster, on the surface, than the TP-Heavy solutions that require synchronization between sites. But, TP-Heavy solutions provide many sophisticated procedures for dynamic load balancing, priority scheduling, process restarts, and pre-started servers that are especially useful for large-scale production environments. These features are the main strength of TP-Heavy products because many of these products have been used over the years to handle thousands of transactions in production OLTP (on-line transaction processing) environments.

In general, small C/S applications are being deployed by using TP-Lite, while large mission critical C/S applications, especially the ones that were "downsized" from mainframe OLTP environments, are using TP-Heavy. In the meantime, many PC LAN-based applications are quite happy with TP-Less. In fact, TP-Heavy is difficult to implement when different vendor DBMSs are involved.

8.11 Additional Information

A large number of URLs give information about different aspects of EC platforms that we have discussed. Exploring these sites is left as an exercise for the reader.

In addition, the following books and articles may be of value (the web articles are indicated by using the [Author/Company Year] convention):

[1] [Cooper 2001], "Merchant Account Tutorial", http://www.iboost.com/profit/selling_products_or_services/getting_started/merchant_accounts/3077.html
[2] [DeFatta 2001] – "Third Party Payment Processors Explained", http://www.workz.com/html/2169.html
[3] [EasyCart 2003] – "Shopping Cart provider", http://www.easycart.com/main.html
[4] Hamilton, S., "E-Commerce for the 21st Century", IEEE Computer, May 1997.
[5] Kalakota, R. and Robinson, M., "e-Business 2.0 -- Roadmap for Success", Addison Wesley, 2000.
[6] Kalakotta, R., and Whinston, A., "Frontiers of Electronic Commerce", Addison Wesley, 1996.
[7] Kay, E., "Compaq wants your Internet business", ComputerWorld, April 12, 1999.
[8] Ketchpel, S. , H. Garcia-Molina, Making "Trust Explicit in Distributed Commerce Transactions", 16th International Conference on Distributed Computing Systems (DCS96), 1996.
[9] Ketchpel, S. , H. Garcia-Molina, "Distributed Commerce Transactions with Timing Deadlines and Direct Trust", IJCAI 1997.
[10] Luftman, J., "Competing in the Information Age: Strategic Alignment in Practice", Oxford University Press, 1996.
[11] Magretta, J., "The Power of Virtual Integration: An Interview with Dell Computer's Michael Dell", Harvard Business Review, March-April 1998.
[12] [Miller 2000] – "Choose Your Shopping Cart", http://www.workz.com/html/1509.html
[13] Özsu, T., and P.Valduriez: "Distributed and Parallel Database Systems. The Computer Science and Engineering Handbook", Allen B. Tucker (Ed.), CRC Press, 1997.
[14] [PaySol 2001] – "Payment Solutions", http://e-commerce.internet.com/reviews/glance/0,,3691_5,00.html,
[15] Shea, P. "New opportunities in ecommerce", Australasian Business Intelligence Jan 24, 2003
[16] Shunter, M., and Waidner, M., "Architecture and Design of a Secure Electronic Marketplace", Procs. JENC8: 8th Joint European Networking Conference, Edinburgh, May 12-15 1997.
[17] Steinauer, D.D., S.Wakid, S.Rasberry, Trust and Traceability in Electronic Commerce. StandardView vol. 5 n. 3, Sept. 1997.

[18] Stylianou, A., et al, "Perceptions and attitudes about eCommerce development in China: an exploratory study", Journal of Global Information Management April-June 2003

[19] [Sussis 1999] – "Accepting Credit Card Payments", http://e-commerce.internet.com/solutions/e-consultant/print/0,,9571_118621,00.html,

[20] Tennenbaum, J., Chowdhry, T., and Hughes, K, "Eco System: An Internet Commerce Architecture", IEEE Computer, May 1997.

[21] Turban, E., et al, "eCommerce: A Managerial Approach", Prentice Hall, 2nd edition, 2002.

[22] Tygar, J., "Atomicity in Electronic Commerce", Proc. of ACM Symposium on Principles of Distributed Computing, May 23-26, 1996.

[23] Umar, A., "Object-Oriented Client/Server Internet Environments" Prentice Hall, February 1997.

[24] Umar, A., "Application (Re)Engineering: Building Web-based Applications and Dealing with Legacies", Prentice Hall, May 1997.

[25] Umar, E., "EC Bedrock", Database Programming and Design, Nov. 1997.

[26] Wardley, M. and Pang, A., Internet-enabling Enterprise Applications, ComputerWorld, April 12, 1999.

[27] [Walsh 1999] – "Understanding Internet Payment Protocols", http://www.nwc.com/shared/printArticle?article=nc/1009/1009ws33.html&pub=nwc