NOTE TO USERS

This reproduction is the best copy available.

UMI®

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

u Ottawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Juan Wang

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Web services Case Study and Implementation in Financial Industry

TITRE DE LA THÈSE / TITLE OF THESIS

David Wright

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Morad Benyoucef

John Nash

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Web services Case Study and Implementation in Financial Industry

By

Juan WANG, B.S.

THESIS
Presented to the Faculty of Management
University of Ottawa
In Partial Fulfillment of the Requirements for the Degree

Master of Science

The University of Ottawa
September, 2005

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-11446-0
Our file Notre référence
ISBN: 0-494-11446-0

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

© Juan Wang, Ottawa, Canada, 2005

Abstract

Web services refer to a family of technologies that can universally standardize the communication of applications in order to connect systems, business partners, and customers cost-effectively through the World Wide Web. Web services will ease the constraints of time, cost, and space for discovering, negotiating, and conducting e-business transactions. As a result, they dramatically changed the way businesses design their applications as services, integrate with other business entities, manage business process workflows, and conduct e-business transactions.

The purpose of this thesis is to investigate the current state of Web services technology and to evaluate the potential effectiveness of this technology for financial industry. By conducting several case studies, the advantages of Web services implementation are discussed, including financial areas and other related E-businesses. Non-technical aspects of Web services such as the value added, cost reduction, implementation contraction, project reusability, and business expanding are discussed as well. To demonstrate the use of Web services for financial industry, three Web service application prototypes were built within different domains relevant to international trading procedure. This simulation project is judged to be a successful demonstration of the potential applications of Web services for financial industry.

Table of Contents

Part I: Introduction

In recent years, Web services have become a very popular issue not only in the IT industry but also in Business. A Web service is an application or information resource that can be accessed using standard Web protocols. It is a "Plug and Play" application and applicable to any type of web environment: Internet, intranet and extranet. As the need for application-to-application communication and interoperability grows, Web services is expanding rapidly, and provides a standard means of communication among different software applications.

In this section, we introduce Web services in terms of features, advantages and weakness compared to other technologies, and also review the current development of Web services and different tools provided by several main software venders.

To facilitate business tasks, enterprise applications must communicate with one another and share data. Historically, this was accomplished through proprietary specifications and data formats. However, the emergence of the World Wide Web and XML—an open standard for data exchange—has increased the possibility for interoperable system-to-system communications. Web services are software programs that use XML to exchange information with other software via common Internet protocols. Basically, a Web service communicates over a network to supply a specific set of operations that other applications can invoke. This means that an application residing on one computer can send requests to, and possibly receive responses from, applications on other computers.

"A Web service is a software system identified by a URI (Uniform Resource Identifier), whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."1

Representing a new generation of web-based software component methodology, Web services are modular, self-contained, self-describing software components. They are published and consumed across the web, and can be easily located and "checked out," online and dynamically, using UDDI and corresponding search engine. As a result, Web services could be seen as web-centric for application developers.

In general, Web services have the following main characteristics:

- A Web service is accessible over the web by using platform-independent and language-neutral web protocols that ensure easy integration of heterogeneous environments.
- A Web service provides an interface that can be called from another program. This kind of application-to-application interface can be invoked from any type of application client or service. The Web service interface also acts as a liaison between the web and the actual application logic that implements the Service.
- A Web service is registered and is located through a Web service Registry, which enables service consumers to find services that match their needs.
- Web services communicate by passing messages to each other and connect systems loosely. A layer of abstraction is added between the Web service interface and its environment to make connection flexible and adaptable.

1 Motivation

The financial services industry has always been early in finding and adopting beneficial uses for new, leading-edge technologies. As such, there would no surprising to find that financial services companies also surface to take the lead in implementing Web services in the near future. SWIFT (the Society for Worldwide Inter-bank Financial Telecommunication) is experiencing a new evolution of message service caused by World Wide Web. The evolution includes message standard changing from MTs (Message Types) to XML-based messages and network transition from X.25-based network to SWIFTNet. As dominating the inter-bank communications market, SWIFT starts using Web services integration to provide complex services to more customers from 2002. In order to cope with this new challenge, the SWIFT community which includes banks, broker/dealers and investment managers, as well as their market infrastructures in payments, securities, treasury and trade has the demand to exploit the new technology to attract more export customers , improve their business agility, and achieve the automation of the international trade process.

2 Contribution

This thesis will explore some of the foreseeable, beneficial implications of Web services in the financial services industry by way of four sample case studies, and offer specific ideas for how financial institutes might apply Web services as a means of solving and improving real-word international import/export problems and processes.

By implementing the international import/export trading system, Banks can broaden its role in the L/C process. Financial institutions provide capabilities to automate the L/C process by using Web services. L/C data described using XML can be passed electronically, in the appropriate format, to be received directly by the target systems. Because its format can be flexibly changed by XML technology, the system can offer services that are easily adaptable to changing market conditions, which helps to attract new customers and improves financial institutions' business agility. The messaging protocol is based on SOAP. This standardizes the connections among all participants in international trade area. In true, we hope to transfer these ideas and encourage similar thoughts into financial industry.

Moreover, the thesis presents how traders could benefit from Web services by reduced warehouse costs and better customer relationships due to prompt delivery of goods. Web services solutions enable exporters to pre-book less expensive shipping methods and insurance policies, because they could know precisely when the goods will be ready to leave the warehouse. Other process participants such as the shipping and insurance companies will also benefit by using Web services as it reduces the cost, time, and errors associated with the international trading process.

By implementing the international import/export trading system, it demonstrates to the Financial Services Industry that such an application can be effectively implemented using Web services. Use of Web services allows businesses to link systems more easily than before, and make services more attractive as Web services are able to develop new products and services quickly and efficiently t. Thereby, Web services are considered the core technology for next-generation e-business.

3 Organization of the thesis

The reminder of this thesis is organized as follows. Part I introduces to the motivation, contribution and organization of this thesis. Part II gives an overview of Web services in terms of components, architecture, advantages and weakness, and its current situation. Part III conducts three case studies within financial industry. Through conducting these case studies, it compares some other likely technologies with Web services and analyzes the business impacts from them in terms of cost reduction, implementation contraction, project reusability, and business expanding. Part IV analyzes the complexity of international import/export trading process, and then drives the implementation for International Import/Export Trade. Three Web service application prototypes were built: Letter of Credit Application Web service, Shipping Quote Web service and Insurance Quote Web service.

Part II: Background of Web services

On the technical level, the reason that Web services are such operational tools that any systems developed by any programming language on any platform can communicate efficiently, is because Web services employ XML (Extensible Markup Language). The link between XML and Web services is inextricable and incontrovertible. All data transfer to or from a Web service always has to be in the form of an XML document. In addition, Registering and announcing one's Web services involve in the following protocols:

- SOAP (Simple Object Access Protocol)
- WSDL (Web services Description Language)
- UDDI (Universal Description, Discovery and Integration)

These technologies enable communication among applications in a manner that is independent of specific programming languages, operating systems and hardware platforms. SOAP provides a communication mechanism between services and applications, WSDL offers a uniform method of describing services to other programs and UDDI enables the creation of searchable Web services registries. When deployed together, these technologies allow developers to package applications as services and publish those services on a network (Figure 1).

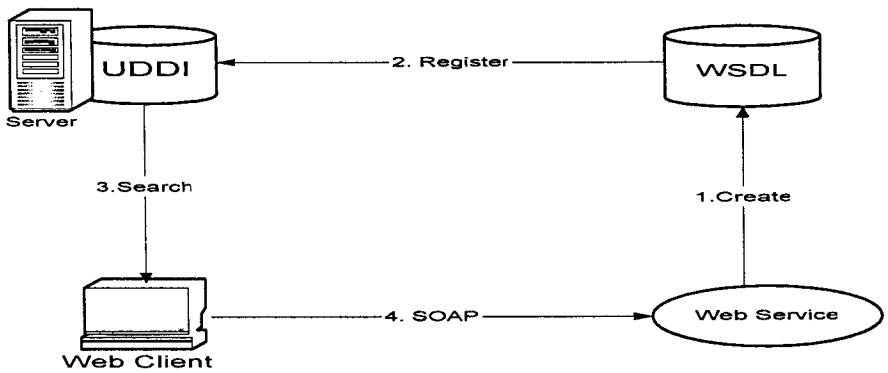


Figure 1: Web Services Workflow

XML

Unlike many technologies which begin as proprietary solutions and become standards, XML was defined by the World Wide Web Consortium (W3C) as an open, standard technology. XML is a widely accepted standard for describing data and creating markup languages.

Data Independence, or the separation of content from its presentation, is the essential characteristic of XML. Because XML documents describe only data, any application that understands XML- regardless of the application's programming language or platform – has the ability to format XML in a variety of different ways. Recognizing this, software developers are integrating XML into their applications to improve Web functionality and interoperability. XML documents contain data, but no formatting instructions, so applications that process XML documents must decide how to display the documents' data.

As applications become more Web enabled, it seems likely that XML will become the universal technology for representing data passed between Web applications. All applications employing XML will be able to communicate, provided that they can understand each other's XML markup, or vocabulary. This high level of interoperability makes XML an ideal technology to enable Web services, which communicate among systems without regard to operating systems and hardware platforms. The core standards enabling Web services, which are described in the next three sections, are based on XML.

SOAP

SOAP in one of the most common standards used to deliver Web services. The purpose of SOAP is to enable data transfer between systems distributed over a network. When an application communicates with a Web service, SOAP messages are the most common means through which the two systems exchange data. A SOAP message sent to a Web service invokes a method provided by the service, meaning that the message requests that the service execute a particular task. The service then uses information contained in the SOAP message to perform its function; if necessary the Web service returns the result via another SOAP message.

A SOAP message consists of three main parts: an envelope, a header and a body. The envelope wraps the entire message and contains the header and body elements; the header is an optional element that provides information regarding such topics as security and routing. The body of the SOAP message contains the application-specific data that is being communicated. The data is marked up as XML and adheres to a specific format, which is defined by the Schemas we mentioned earlier. This formatting enables the recipient to process the data correctly. SOAP messages are received and interpreted by SOAP servers, which, in turn, trigger Web services to perform their tasks.

WSDL

Another standard that plays a crucial role in enabling Web services is WSDL. It is an XML-based language through which a Web service can convey to other applications the methods that the service provides and how those methods can be accessed. A WSDL document defines the kinds of messages a Web service can send and receive, as well as specifying the data that a calling application must provide for the Web service to perform its task. WSDL documents also provide specific technical information that informs applications about how to connect to and communicate with Web services over HTTP or another communications protocol. Nearly every Web service published on the Internet is accompanied by an associated WSDL document, which lists the service's capabilities, states its location on the Web and provides instructions regarding its use.

Nevertheless, it is important to realize that WSDL is a language meant to be read by applications, rather than by humans. Although the structure of WSDL documents might appear complex, computers that understand WSDL can process the documents and extract the information they need. Furthermore, most Web services development tools generate WSDL documents automatically. This means that, if a programmer develops a Web service, the software used to build the service creates an appropriate WSDL document for that service. Therefore, it is not necessary for developers to under stand the syntax of WSDL fully when building and deploying Web services.

UDDI

UDDI enables developers and businesses to publish and locate Web services on a network. As its name implies, the specification allows companies to describe their own services and electronic processes, discover those of other companies and integrate others' services into their systems. UDDI defines an XML-based format in which companies can describe their electronic capabilities and business processes; the specification also provides a standardized method of registering and locating the descriptions on a network, such as the Internet. Companies can store their information either in private UDDI registries, which are accessible only to approved business partners, or in public UDDI registries, which any interested party can use. The largest, most comprehensive public UDDI registry is the UDDI Business Registry (UBR), which was developed to facilitate the formation of new business relationships.

4 Architecture of Web services

There are 5 basic components of Web services in its architecture:

- Service-Oriented Architecture (SOA)
  This is the basis of the Web services model. "Similar to object-oriented model, SOA treats all entities as services, and defines the service model and enables

services residing on any network to be published, located and invoked by other services. SOA has some characteristics: service-abstraction, service-encapsulation, service-modularity and service-polymorphism."[2]

- Service roles

  In SOA, Web services play different roles, such as service provider, service broker and service requestor.

- Technology stack

  Technology Stack provides a high-level view of SOA. It consists of those standards we mentioned before: SOAP, WSDL, UDDI and ebXML. Figure 2 depicts a conceptual overview of the SOA stack.

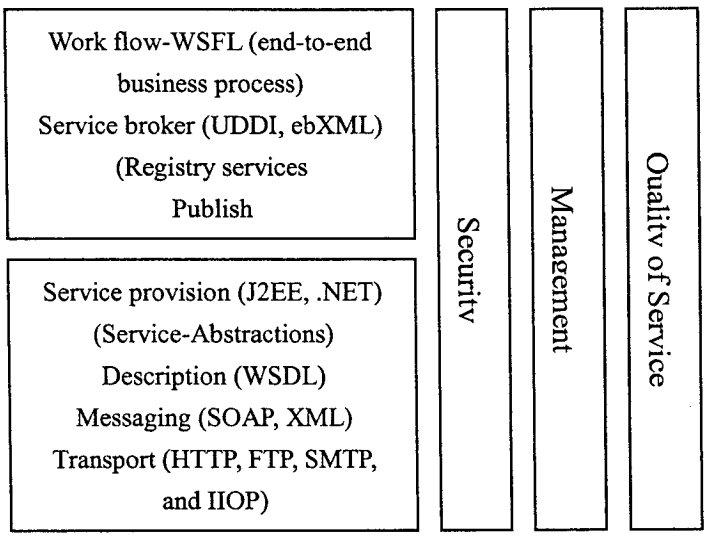


Figure 2: SOA Stack

- Architectural processes

  In the Web service model, there are primary processes: description, discovery and invocation.

- Value chain

  Web services seem to be a very promising proposition for leveraging investments for a better Return On Investments (ROI). Value chain of a Web service is a combination of the service supplier and other suppliers together to add value to the service and satisfy market demands.

5 Advantages

Web services dramatically and positively alter everything related to applications, both on the supply and demand side. It is not hard to conclude some advantages associated with Web services.

- Web services operate using open, text-based standards, which enable components written in different languages and for different platforms to communicate.
- Web services promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Web services are comparatively easy and inexpensive to implement, because they employ an existing infrastructure (a network, such as the Web) to exchange information. Moreover, most applications can be repackaged as Web services, so companies do not have to adopt entirely new software.
- "Web services can significantly reduce the costs of enterprise application integrations (EAI) and business-to-business (B2B) communications, thus offering companies tangible returns on their investments."[3]
- Web services can be implemented incrementally, rather than all at once. This lessens the cost of adopting Web services and can reduce organizational disruption resulting from an abrupt switch in technologies.

6 Weaknesses

Web services offer many benefits, but also create significant challenges for application developers and IT Staffs. Although SOAP v1.2 was standardized by W3C in 2003, the key problem is the WSDL and UDDI –the main standards that drive Web services—are still in draft form. SOAP and WSDL are under development by the W3C, and UDDI has not yet been submitted to a standards organization. This means that the protocols and specifications are likely to change in the near future. Many businesses want to wait until the underlying technologies are stable before adopting Web services. Others are nervous that the current cooperation among software vendors such as Microsoft, IBM and SUN to create interoperable implementations might fail, resulting in splinter standards and incompatible Web services implementations.

Other Web services challenges involve defining and guaranteeing Quality of Services (QoS). Before invoking a Web service, customers often want to verify that service will meet their expectations. Possible QoS problems with Web services include slow response times, and an inability to handle large numbers of requests. The overall performance of Web services depends on application logic, network, and most importantly on underlying messaging and transport protocols, such as SOAP and HTTP. The SOAP protocol is still maturing and harbors a lot of performance and scalability problems. The SOAP protocol uses a multi-step process to complete a communication cycle. The SOAP request begins with the business logic of your application learning the method and parameter to call from a Web services Description Language (WSDL) document. This whole process is a time-consuming one, which requires various levels of XML parsing and XML validation and hence

hits the performance of the Web service. Some businesses have developed service-level agreements (SLAs), contracts between services providers and consumers that guarantee certain amounts of uptime, performance, security and so on. Also, independent companies have begun to provide third-party evaluations of Web services.

In general, although Web services technologies offer important new computing capabilities, software vendors and service providers still need to work out many details regarding Web services interactions. Payment plans, standards for service quality and contracts that guarantee levels of quality are important factors that will affect the affect the adoption of Web services.

7 The Current Situation of Web services

Web services encompass a set of related standards that can enable any two computer applications to communicate and exchange data via the Internet. Although the true impact of Web services is not yet known, many factors—including software vendors' widespread support for the technology – indicate that Web services will radically change IT architectures and partner relationships.

Web services take advantage of object-oriented programming techniques in that Web services enable developers to build applications from existing software components. This can greatly reduce the effort required to implement certain kinds of systems. In addition, because Web services operate using open standards, it enables any two software components to communicate—regardless of differences in programming languages or platforms. As the result, Web services improve distributed-computing capabilities by addressing the issue of limited interoperability.

Web services are comparatively easy and inexpensive to implement, because they employ an already existing infrastructure (a network, such as the Web) to exchange information. Also, Web services significantly reduce the costs of enterprise application integration (EAI) and business-to-business (B2B) communications.

Challenges to Web services are that SOAP, WSDL and UDDI are still in draft form; Web services lack standard security procedures; and they involve defining and guaranteeing Quality of Service (QoS).

However, when a manager chooses a Web services solution provider, he or she should take two steps. First, he or she must match Web services products to the company's requirements, such as legacy system, developing team, budget, time, and so on. Second, the manager must determine the level of vendor support offered to

clients. For example, some companies might feel more comfortable trusting a smaller vendor that offers more personalized service and support. Other organizations might prefer the support the product variety provided by a large, established vendor like IBM, Microsoft, or SUN. Accordingly, let us introduce several main Web services vendors briefly.

7.1 Web services platforms, Vendors and Strategies

Many software vendors have created Web services platforms, which consist of programming tools with which developers construct and deploy Web services and server software. Web services platforms "hide" many of the programmatic details and enable programmers to create and deploy Web services easily.

Microsoft and the .NET Platform

"At the early stage, Microsoft is one of the dominant companies in the Web services market—some estimates consider the company to be almost a year ahead of others in the development of Web services technologies."[4] Visual Studio .Net enables programmers to design Web services in a variety of languages, including C++, C# and Visual Basic.Net. However, ".NET technologies are available only for Windows 2000 and XP. For Windows developers working in older Windows platforms or Windows 2000/XP systems that do not have .NET, Microsoft created the Microsoft SOAP Toolkit. Instead of maintaining My Services data in central registries, Microsoft is packaging the Web services technology with other software such as Windows XP. This enables companies to maintain their own secure data repositories."[5]

IBM

"IBM incorporated Web services technologies into allots middle ware applications, including WebSphere, DB2, Lotus and Tivoli."[6] IBM also has created a technology for designing and deploying Web services, called the Web services Toolkit. The WebSphere Application Server is IBM's main deployment platform for web-based applications. Up to now, IBM has released a set of tools that further support Web services development on WebSphere:

- WebSphere Studio Application Developer allows developers to build, test and deploy J2EE applications, including Web services.
- WebSphere Studio Site Developer contains a Web services development environment, including a private UDDI repository, to create, manage and maintain interactive applications.
- "Enterprise developer for multiplatform is a development environment that offers

integration of Web services in complex e-business systems."[7]

Besides, IBM's enterprise database, DB2, also has integrated Web services technologies. "Lotus Notes and Domino also provide support for Web services standards such as XML, SOAP and UDDI."[8] "IBM's Tivoli network-management software incorporates Web services technology via the Web services Manager—an application that monitors the performance of Web services transactions – and the Secure Way Policy Director—an application that provides security for Web services transactions."[9]

Sun Microsystems

Sun Microsystems's Web services strategy is based on the SUN Open Net Environment (Sun ONE), which consists of three components—a vision, an architecture and a conceptual model for developing standards-based software. "Sun also considers support from solutions consultants and collaborative programs to be another essential part of the strategy."[10]

The Sun ONE platform includes three products: the Solaris Operating Environment, The Infrastructure Software (formerly iPlanet Web applications) and the SunONE Studio (formerly Forte). The Sun ONE allows programmers to deploy Web services using third-party products. By integrating disparate products, programmers can develop Web services infrastructures that best suit their companies' requirements. "Sun ONE incorporates support for open standards, including SOAP, J2EE, UDDI and ebXML, to help ensure high levels of interoperability and system integration."[11] The Sun ONE vision incorporates a model for software development, in which critical business information and applications are available at any time to any type of device, including cell phones and PDAs. "Sun ONE's goal is to help developers create networks of distributed applications or Web services that are highly reliable and promote the reuse of components and services."[12]

Major software vendors such as IBM, Microsoft, SAP, SUN, and Oracle are all embracing Web services standards and are releasing new products or tools that are Web services enabled. Web services will ease the constraints of time, cost, and space for discovering, negotiating, and conducting e-business transactions. As a result, Web services will change the way businesses design their applications as services, integrate with other business entities, manage business process workflows, and conduct e-business transactions.

7.2 Opportunities and Challenges

Web services refer to a family of technologies that can universally standardize the communication of applications in order to connect systems, business partners, and customers cost-effectively through the World Wide Web. The emerging Web services standards and technologies make it possible to provide software functions and business services over the Web and to integrate companies with trading partners seamlessly. Web services' supporters believe that they are "the" technology to bridge the gap between IT and business. In addition, more and more complex business problem are expected to be handled and solved by Web services technologies in near future, which will attract and create more and more interest from B2B electronic commerce and mobile commerce, as well as enterprise application integration (EAI).

Web services also can decrease the complexity and cost of integrating applications, or connecting them to enable direct communication and information exchange. "Forrester research estimates that, without using Web Serves, the average company spends up to one million dollars to research, test, implement and maintain software that can link two separate applications over the Internet."[13] By contrast, Web services facilitate less expensive and more flexible integration solutions—especially when connecting applications written in different languages for different platforms. Companies can employ Web services to integrate their own applications or to integrate with suppliers, distributors, partners and corporate clients.

Web services can improve corporate software development by reducing the time and expense involved in developing a software application. With companies adopting Web services, they are beginning to develop private UDDI or ebXML registries, which catalog the Web services maintained by a company or a group of partner companies. Instead of designing software" from scratch," programmers can use registries to locate existing Web services, and then incorporate those services into applications. Programmers can find additional Web services by searching public registries, such as the UDDI Business Registry. A programmer might compile an e-commerce application from numerous publicly and privately held Web services, such a Web services that processes credit-card payments and a Web service that provides driving directions to the company's closest store. Since the developer does not have to create new software to perform these tasks, the application can be completed faster and at less cost. "According to Gartner, Web services could increase the efficiency of software-development projects by up to 30 percents."[14]

Realizing these advantages, many organizations are beginning to adopt Web services. Figure 3 summarizes the results of a 2002 Jupiter Media Metrix survey, in which companies were polled regarding their plans to deploy Web services over the next year. "The survey indicated that only 23 percent of companies had no plans to adopt Web services."[15]

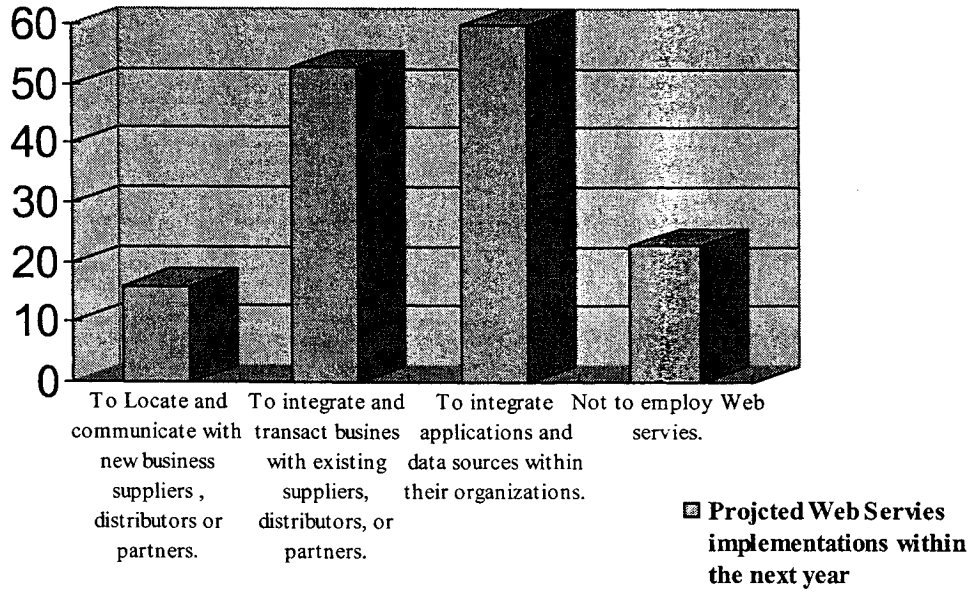


Figure 3: Survey Results from 2002 Jupiter Media Matrix

However, Web services are not an ideal solution for every software-development project. As we know, Web services security and QoS protocols are not fully developed. "Some experts believe that Web services should not yet be used to transmit highly confidential or business-critical data across organizational boundaries."16 Also, "Web services are not recommended for systems that handle a large number of transactions, because processing SOAP messages can impede performance."17 "Companies should implement Web services slowly, beginning with internal integration projects or services that are shared among trusted partners. Once Web services and security standards mature, companies might deploy Web services that are more widely accessible."18

Part III: Case Study

After we get basic concept of Web services, we are continuous to conduct several case studies in this section, including Web services implementation examples from CIBC, Standard & Poor's and SWIFT. These selected cases come from typical and representative industries that are encountering the innovation caused by Web services. In certain extent, they bring wide affects on and could be examples for our Import/Export International Trade Web services implementation. Through these analyses, we acknowledge the precession of Web services application's designing,

developing, and deploying, thoroughly understand the benefits it bring, and help us to develop our Import/Export International Trade project.

More, the cases we chose come from main software vendors in the current market, including Microsoft, IBM, and Sun Microsystems, make us look insight of those technology and features.

8 Canadian Imperial Bank of Commerce CIBC

As one of the largest banks in Canada, CIBC provides financial services to more than nine million customers, including retail and small business banking customers as well as corporate and investment banking customers. CIBC has approximately 1,100 branches and offers customers the largest ABM network in Canada, with more than 4,400 ABMs in branch and non-branch locations across the country.

Like many enterprises, CIBC is facing today with the dilemma of responding to an ever growing demand to serve Intranet and Internet inquiries against its enterprise information systems and reduce their overall IT expenses as well. Because Web services technology has been thought as an enabler that can help define and integrate loosely coupled software services, CIBC has begun an investigation into the use of Web services as an application integration model.

The bank found they needed a strategy for application integration between lines-of-business, which asks platform and language independent strictly. After performing a research by CIBC Corporate Architecture team, they realized that the bank's overall strategic architecture needed to include an integration tier, separate (or de-coupled) from an application layer, allow legacy assets to be served as "coarse-grained" business services to middle-tier application servers, essentially providing a set of reusable "Core Services" that spanned across lines-of-business.

With the help of the IBM jStart Team, a proof of concept was defined to address the following requirements:[19]

- Evaluate the ease of exposing Legacy system assets to application tiers via an open standard interface such as WSDL and SOAP.
- Provide a basis for examining the usage of IBM Websphere as an edge server for CICS and IMS transactions utilizing Web services.
- Test a means of integrating some "Core Services" such as the customer account data, which can be utilized across lines-of-business.
- Test the ease of use of IBM Websphere Studio Application Developer tool relating to the creation and deployment of Web services.

Working with the IBM jStart Team, CIBC developed a three-tier solution. "The first tier, associated with the end-user, pertained to client software, which required only a web browser. The middle-tier development environment was IBM's WebSphere Application Server, along with various components including an EJB that represented the Web service client proxy. In the third tier or backend, IBM's WebSphere Application Server for OS/390 was deployed to connect to an existing instance of IMS. Installed on the application server was the Customer Account Data Service along with the associated business component that handled the communication with IMS, namely EBCDIC to ASCII transformations."[20]

At the result, CIBC was able to demonstrate that Web services technology existed today to help them integrate enterprise applications to legacy assets using open standards. "CIBC has built on its initial experience with Web services, and is going to investigate a number of follow-on activities, such as: an evaluation of JCA enhancements in IBM's Web services development tools; Performance and capacity measurement tools; and applicability of UDDI in their strategic architecture."[21]

Web service is a rapid emerging technology that could address many concerns on real-time data integration and analysis in financial industry, such as trading and risk analytics, pre-trade decision support, and real-time portfolio balancing. They could provide access to computing resources for internal users and customers, regardless of location. Despite of using Web services internally, CIBC benefits from the Web services integration in terms of reducing developing and operating time, lowering costs, changing the way banks use distributed computing, and enabling the delivery of operational applications and line-of-business applications.

"Celent expects the expenditure for Web services and XML technologies to increase to US$1.4 billion by 2007 in the US financial services industry."[22] With the Web services deployment expanding, more and more developing professionals in financial services believe Web services will provide a robust integration infrastructure that can handle the actual transportation of information and orchestration of business processes throughout the entire financial supply chain, and enable financial institutes to economically take advantage of new business opportunities by automating manual processes for leases, loans, lines of credit, credit cards, and so on.

9 Standard and Poor's

Standard and Poor's (S&P, www.standardandpoors.com), a division of the McGraw-Hill Companies, provides financial information, industry analysis and data on fiscal performance and trends for the financial-services industry. "The company's main goal is to offer outside, objective resources that help individuals and

corporations make well-informed investment decisions."[23] In order to assess the performance of companies and investments, managers and business executives require the most up-to-date financial reporting possible. Thus for S&P, the critical aim is to constantly seek out new ways to keep its clients informed. In 1998, S&P began offering some of its printed material online – the company soon realized that the Web could vastly improve its ability to deliver financial information to clients.

At the beginning stage of Web evolved, S&P has tried some new ways to delivery its information and services, including the use of Web services technology. Initially, the company developed more than 60 Web services, and packaged all Web services completely, so interested customers had to purchase all 60 services at once. Thinking about this inconvenience, S&P changed to provide each Web service separately and enable customers to pick and choose their specific applications. "S&P maintains over 40 Oracle databases to contain the services' information, and each database is updated regularly to include current and accurate financial data."[24]

"S&P use the Sun ONE platform to create each application as a modular J2EE component. The component is packaged in a SOAP envelope and described using WSDL. The SOAP envelope then is passed to an internal directory, from which other business units within the company can access the information. These components also can be published externally, which allows client businesses to incorporate the services on their own Web sites or portals."[25] Examples of available S&P Web services include a quote engine, rules-base screening tools, equity-quote services and financial calculators.

Through Web services, S&P increases its abilities of providing more current data and value-added services to customers. Web services also allow S&P to provide more current data and value-added services to customers, consequently attract new customers and expand into new markets without significantly increasing the company's infrastructure. Certainly, S&P's Web services are available worldwide via the Internet, and international customers can access up-to-date financial reports and industry data, regardless of location.

10 SWIFT

As its name implies, SWIFT, the organization known as the Society for Worldwide Inter-bank Financial Telecommunication, has been focused for most of its existence on facilitating the flow of communications between banks. It has been supremely successful, transforming a telex-based, paper-intensive process into a fully electronic network linking together many thousands of banks in nearly 200 countries.

SWIFT dominate the inter-bank communications market to the extent that there is really no competition. The SWIFT community includes banks, broker/dealers and investment managers, as well as their market infrastructures in payments, securities, treasury and trade. All banks worldwide depend on SWIFT to use its message formats for international business through its network.

10.1 SWIFT and STP

SWIFT is a worldwide community of financial institutions, whose purpose is to be the leader in communications solutions enabling interoperability between its members, their market infrastructures, and their end users. It is a powerful support to financial institutions from all over the world. These institutions are active in payments, securities, treasury, and trade services. They collectively exchange millions of messages valued in trillions of dollars every business day across the SWIFT messaging platform. SWIFT has played two key roles in its transformation process, which are the network provider and the standards authority for the formats of the messages exchanged over the network.

Because of these significant roles in international banking domain, SWIFT strictly requires its messaging system should be straight through processing (STP), secure and reliable, and cost-effective.

10.1.1 What is the Straight Through Processing (STP)?

Straight Through Processing (STP) is a solution that automates the end-to-end processing of transactions for all financial instruments from initiation to resolution, which is set to revolutionize the financial industry. STP is the initiative to reduce or eliminate any manual handling of a financial or a trade transaction, and aims to improve automation of the entire processing chain to achieve greater efficiency, timeliness and cost-effectiveness. STP solutions are needed to help financial markets firms meet the move to one-day trade settlement (T+1), as well as to meet the global demand that has resulted from the explosive growth of online trading.

STP can increase the automation of the entire processing chain. It encompasses a set of applications, business processes, and standards that will redefine the settlement and process within the capital markets industry.

There are several key benefits of STP:[26]

- Better electronic connectivity among different entities involved in the trading cycle.
- Integration of front, middle, and back office applications based on standards.
- Elimination of a lot of manual activities and redundant processes in the

end-to-end processing of trade transactions.

- Higher accuracy of trade execution and settlement.
- Reduced operational costs.
- Reduced trade cycle.

SWIFT defines STP as "the handling of messages without resort to manual intervention or message repair, except for reasons of policy". Definitions given to SWIFT by its customer banks during STP Reviews vary, including "intervene as little as possible, receive the minimum of enquiries possible", "using your hand only once - preferably the customer's hand" and "straight through from customer to customer".[27]

In order to execute STP, the two critical goals of SWIFT message system should satisfy with the following requirements.

- Quantity - Real-Time Information

The whole bunch of trade information passed between the buyer, seller, exchangers, and any other participants involved in the trade processing should be on a real-time basis at fast speeds, which requires either synchronous or asynchronous integration between applications running on different platforms.

- Quality – Message Security

The key of STP executing is secured interoperability for both internal STP and external STP as well. For external STP, it is obviously more risky because it involves the use of the Internet or Virtual Private Network (VPN*). External STP enables communication cross boundary between enterprises, meanwhile financial companies must protect their internal network against malicious attacks through these open ports. Also, the data transmitted over the Internet or any other network has to be secured and can not be left unguarded.

10.1.2 Why use Web services for STP?

Being able to achieve seamless trade processing, SOA (Service-Oriented Architecture), one of basic components in Web service architecture, provide the foundation and the application architecture for Web services. A SOA-based framework is capable of providing support for multiple XML standards at the same

---

* a *network* that is constructed by using public wires to connect nodes. For example, there are a number of systems that enable you to create networks using the Internet as the medium for transporting data. These systems use encryption and other security mechanisms to ensure that only authorized users can access the network and that the data cannot be intercepted. VPN solutions support remote access and private data communications over public networks as a cheaper alternative to leased lines.

time, and adding additional standards support without significant redevelopment effort.

There are several benefits of using Web services as one of the core technologies for STP. As we will see below, the main benefits are:

- **Based on open standards**

Web services are very flexible and can fully utilize open standards. There are some principal standards for the financial industry that will enable STP for all financial instruments.[28]

- FpML (Financial products Markup Language), based on XML, aims to standardize e-commerce activities in the field of financial derivatives, swaps, and structured products. Figure 4 [29]shows an example of FpML. FpML is used between participating companies for communicating over-the-counter (OTC) transaction details, within a company for the purpose of sharing OTC transaction information, and between a participating company and an outside firm offering a service related to the OTC transaction. FpML is freely licensed and, because it is independent of the software or hardware used by participating companies, ensures interoperability.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <FpML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="file:///C:/
Documents%20and%20Settings/Brian/My%20Documents/FpML/valuation/
valuation_mk2.xsd">
- <!-- the message header (not expanded) -->
<header />
- <!-- the trade information (not expanded) -->
<trade />
- <!-- Parties are abbreviated-->
<party id="abc">ABC Bank</party>
<party id="def">DEF Bank</party>
- <!--the returned value-->
- <valuationReport>
<name>Swaption Value</name>
- <valuation>
<receiverPartyReference href="def" />
<payerPartyReference href="abc" />
<currency>USD</currency>
<npv>121.00</npv>
<valDate>2003-06-04</valDate>
<npvSide>ask</npvSide>
</valuation>
</valuationReport>
</FpML>
```

Figure 4: Example of FpML

- FIX (Financial Information Exchange protocol) is a language that defines specific kinds of electronic messages (pre-trade and trade messages) for communicating securities transactions between financial institutions, primarily investment managers, brokers/dealers, and stock exchanges. FIX is a public-domain specification owned and maintained by FIX Protocol, Ltd. The most important feature of FIX that differentiates it from other protocols in the financial industry is that FIX is a connected, session-based protocol. Figure 5 depicts an example of FIX .[30]

have some Web services implementations as part of their STP. That's primarily internal, and not across the full trade lifecycle, but Web services represent a very common integration tool because of easier integration. STP is largely about integration and workflow, and Web services are a great way to integrate messaging and workflow. It's a natural tool for STP."[32]

- **Better and Cheaper Customer Services**

Both user-centric and application-centric Web services can play a major role in customizing a range of financial and non-financial product packages suited for each customer's specifications and making it cheaper and faster to deliver. For example, "Washington Mutual (WAMU) used a Web service approach to rapidly deploy nearly 20 customer-access sites. The Web services standardize the flow of information and processing between WAMU's clients and third-party serves providers, such as property appraisers and credit-rating agencies. Clients can obtain near-instant credit decisions for consumer loans, and it streamlines the mortgage approval process. For WAMU, Web services help them speed to market, lower developing cost, and eventually achieve client satisfaction."[33]

- **Cost-efficiency**

We have analyzed above real examples of Web services implementations to verify absolute competition of Web services. Obviously, the cost-effective is a key feature of Web services. It decreases complexity and enable to be re-usable, thus reduces the cost and time required for integrating different systems on different platforms.

As the efficient solution for enterprise integration, Web services plays a very significant role for financial industry to deliver STP. In a BEA survey of the top 100 US-based financial institutions, 85% of respondents said that Web services would play a role in STP within the next two years.[34]

Although Web services is one of the technologies used for STP, as the competitive solution, Web services will help banks to transit into the nest generation of STP, where customers can access bank's services automatically. Web services have capabilities to help out innumerable financial institutions like the SWIFT, to seamlessly integrate their own legacy systems. The result is an industry-wide STP that gives the opportunity to even the smallest buyers for real-time connectivity to core services and utilities.

10.2 SWIFT and XML

As the critical messaging provider, SWIFT experienced several evolutions of message types, from initial Telex message, then Message Types (MTs), until creating ISO 15022-compliant MTs. However, with the information explosion caused by the Internet, most organizations have invested in an architecture based on up-to-date technology, such as IP technology for their network, PKI technology for their security and XML technology for their communications, SWIFT started consulting its users to determine their requirements in this evolving business environment in 1997. Faced with the industry's needs for structured communication and recent technological progress, combined with the increased availability of a broad and reliable commercial offering, the options for SWIFT were either to enhance the MT syntax to meet this need for more complex communication, or to address these new needs by complementing the MT syntax with other more versatile syntaxes, such as XML.

After comparing MT and XML syntax, SWIFT found the only difference in the area of information transportation for financial transactions is that XML has additional features:

- XML is a formal language. Rather than taking charge of repenting the content of a message, XML also describes the exact structure of the message, for example, the format of each field, either optional or mandatory, the sequence of fields, and so on. XML is also called a specification language. The structure of MT message lacks of a computer readable specification language.
- XML is also the basis of Web services which can be adopted and understood by special programs, automatically linking them into existing applications without human intervention, and eventually meeting the objective of expending or offer new business opportunities.

Moreover, it is very clear that XML could bring added value to SWIFT. If SWIFTStandards XML development approach is defines and uses, XML would be used as syntax of the new SWIFT message types, and easy to generate XML schemas automatically for the message model. Additionally, new XML message types can be deployed on the SWIFT network without any software development, which causes a significant saving for the implementation in SWIFT. In the aspect of extending SWIFT's capability to further support the community, XML brings new opportunities to build new tools and services, for instance, Web services, and XML-based new standards enable faster implementation on SWIFT interfaces with cost reduction.

Prospecting XML in SWIFT trade services

In the trade services area, SWIFT currently offers two sets of FIN-based standards– standards for documentary collections(MT 4xx) and standards for letters

of credit and guarantees(MT 7xx), which are used to meet the bank-to-bank communication needs for the various trade services instruments.

In Sibos conference October, 2004[35], SWIFT proposed to develop a new set of XML-based standards. Figure below gives an example of XML schema and instance that are used in SWIFT. As a new set of XML-based standards developing, SWIFT will experience an inter-bank communication evolution in the trade services domain. "The SWIFTStandards XML based TSU (Trade Services Utility) standards will enable bank to automatically match data sets under open account trade, to simplify the handling of traditional documentary business and to offer new trade services to customers."[36] (Figure 6)

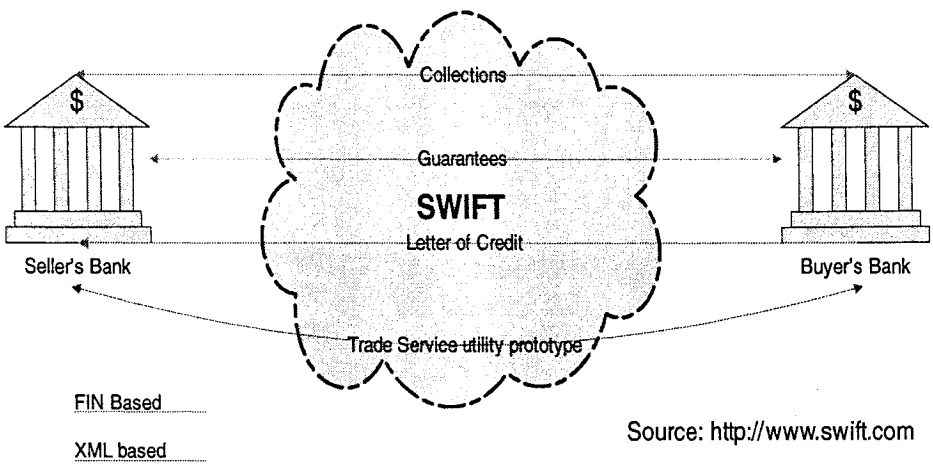


Figure 6: Trade Services Utility Prototype

SWIFT thinks the combination of new SWIFTStandards and the central utility (TSU) will enable SWIFT to broaden services in the trade services market. XML is going to be the syntax of choice, which a majority of institutions, vendors and computer manufacturers is moving, and can bring tremendous savings in terms of implementation, flexibility of use and low cost of related products. "SWIFT intends to offer financial institutions a common language to communicate any kind of financial information between themselves, their market infrastructures and with their clients, on any communication network."[37]

"SWIFT proposes to continue and deepen the dialogue with its community to define the concrete steps on how, when and under which conditions to move further to XML for each business area."[38] The community has chosen XML, and has taken actions to make it a reality. Example of XML message schema and instance are showed as Figure 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Schema version 4.1 - Generated by SWIFTStandards Workstation
(build:R4.1.2.13) on 2003 Aug 07 16:22:42-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iso:xsd:$Message" elementFormDefault="qualified"
targetNamespace="urn:iso:xsd:$Message">
<xs:element name="Document" type="Document"/>
<xs:complexType name="Document">
<xs:sequence>
<xs:element name="Message" type="Message"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Message">
<xs:sequence>
<xs:element name="MessageComponent1" type="MessageComponent1"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="MessageComponent1">
<xs:sequence>
<xs:element name="att1" type="CurrencyAndAmount"/>
<xs:element name="att2" type="ImpliedCurrencyAndAmount"/>
<xs:element name="att3" type="MoneyLaunderingCheck1Code"/>
<xs:element name="att4" type="ISODateTime"/>
<xs:element name="att5" type="SEDOLIdentifier"/>
<xs:element name="att6" type="UKDomesticSortCodeIdentifier"/>
<xs:element name="att7" type="AllOrNoneIndicator"/>
<xs:element name="att8" type="Number"/>
<xs:element name="att9" type="Max35Text"/>
<xs:element name="att10" type="PercentageRate"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="Max35Text">
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="35"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Number">
<xs:restriction base="xs:decimal">
<xs:fractionDigits value="0"/>
<xs:totalDigits value="18"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="AllOrNoneIndicator">
<xs:restriction base="xs:boolean"/>
</xs:simpleType>
<xs:simpleType name="UKDomesticSortCodeIdentifier">
<xs:restriction base="xs:string">
<xs:pattern value="SC[0-9]{6,6}"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="SEDOLIdentifier">
<xs:restriction base="xs:string"/>
</xs:simpleType>
```

```
<xs:simpleType name="CurrencyAndAmount_SimpleType">
<xs:restriction base="xs:decimal">
<xs:minInclusive value="0"/>
<xs:fractionDigits value="5"/>
<xs:totalDigits value="18"/>
</xs:restriction>
</xs:simpleType>
<xs:complexType name="CurrencyAndAmount">
<xs:simpleContent>
<xs:extension base="CurrencyAndAmount_SimpleType">
<xs:attribute name="Ccy" type="CurrencyCode" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:simpleType name="PercentageRate">
<xs:restriction base="xs:decimal"/>
</xs:simpleType>
<xs:simpleType name="ISODateTime">
<xs:restriction base="xs:dateTime"/>
</xs:simpleType>
<xs:simpleType name="MoneyLaunderingCheck1Code">
<xs:restriction base="xs:string">
<xs:enumeration value="PASS"/>
<xs:enumeration value="NOTC"/>
<xs:enumeration value="EXEM"/>
<xs:enumeration value="CLMO"/>
<xs:enumeration value="AUTH"/>
<xs:enumeration value="POEP"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="ImpliedCurrencyAndAmount">
<xs:restriction base="xs:decimal">
<xs:minInclusive value="0"/>
<xs:fractionDigits value="5"/>
<xs:totalDigits value="18"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="CurrencyCode">
<xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Doc:Document xmlns:Doc="urn:isc:xsd:$Message"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<Doc:Message>
<Doc:MessageComponent1>
<Doc:att1 Ccy="EUR">9999999999999.99999</Doc:att1>
<Doc:att2>+0.1</Doc:att2>
<Doc:att3>EXEM</Doc:att3>
<Doc:att4>0001-01-01T00:00:00.001-00:01</Doc:att4>
<Doc:att5>0872317</Doc:att5>
<Doc:att6>SC934543</Doc:att6>
<Doc:att7/>
<Doc:att8>0000000000000001</Doc:att8>
<Doc:att9>Just a text.</Doc:att9>
<Doc:att10>568.85</Doc:att10>
</Doc:MessageComponent1>
</Doc:Message>
</Doc:Document>
```

Figure 7: XML Schema and Instance of SWIFT[39]

10.3 SWIFTNet and Web services

In 2002, SWIFT has signed agreements with integration software providers WebMethods and Cape Clear Software that will allow it to offer its 7,000 member institutions XML-based payments and cash reporting facilities.[40]

Instead of using traditional expensive and complex integration means, Cape Clear enable to integrate with SWIFT network using Web services. SWIFT is migrating its existing X.25-based network to SWIFTNet, which uses Internet Protocol (IP) network technologies and XML.

"The traditional means of integrating applications is difficult, proprietary, expensive, and time-consuming," observed David Clarke, executive vice president at Cape Clear Software. "The ability of Web services to radically alter the economics of integration, while still solving the same problems, is a powerful proposition. We're committed to bringing the benefits of Web services to SWIFT and enabling financial institutions and their clients to access the network and each other quickly, securely, and economically. This technology is already in place and we're demonstrating the Web services enablement of SWIFT at the Sibos conference."[41]

As we mentioned before, the new version of SWIFT messages (XML for SWIFT, as described in ISO15022) are based on the XML standard, which is fundamentally changing the current SWIFT message standards and pushing the movement of SWIFT from X.25 to IP network technologies. Based on business elements relevant to the specific business process, the new message will be easier to be updated and used. Due to embrace the principles of straight-through processing (STP), they also provide more business functionality and expand capacity without any loss of efficiency.

SWIFTNet is an advanced IP-based messaging platform, hosted and managed by SWIFT. It provides a series of products and services for secure and reliable communications of financial information and transaction data. "SWIFT refers to this as a "single window", because once connected, many different services can be used across the single connection".[42]

In order to drive financial institutions success in today's competition environment, SWIFTNet messaging services provide secure real-time and store-and-forward messaging, file transfer and browsing capability, along with the security, reliability, availability and cost expectations of the financial services industry. They can be broken down into three distinct groups: SWIFTNet Interact, SWIFTNet FileAct, and SWIFTNet Browse, shown as Figure 8.

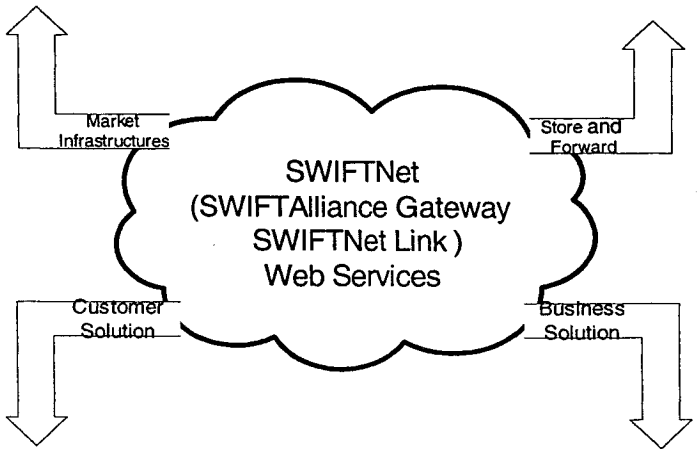


Figure 8: SWIFTNet Web services

Not only providing secure IP network connection, a highly secure and extremely reliable network, SWIFTNet messaging services also offer a wide choice of connectivity options that range from low cost dialup to fully flexible, high bandwidth configurations.

For the purpose of interoperability between customers, SWIFTNet Link adopts Web service technologies to achieve transport, service management and security goals. SWIFTNet Link is used in conjunction with SWIFTAlliance interface Web services, to smoothly integrate user's applications with the SWIFTNet messaging services. "SWIFTNet includes two Web service interface products: SWIFTAlliance and SWIFTAlliance. SWIFTAlliance WebStation is browser based generic interface platform, and is used to access different business solutions over SWIFTNet in an integrated manner. SWIFTAlliance Gateway Web service is an automated application-to-application service and connects to SWIFTNet. SWIFTAlliance Gateway is the interface that offers the single window to all SWIFTNet services".[43]

As end of May 2004, 55% of all BICs (BANK IDENTIFIER CODE) and 44% of the traffic had migrated to SWIFTNet. When the SWIFTNet migration is completed by the end of 2004, the whole community are able to access the new IP-based SWIFTNet business solutions.

10.4 Conclusion

The economics of integrating applications and automating trading relationships is changing. Where previously the investment required to integrate applications was prohibitive to all but the largest institutions, Web services is heralding a new connected business environment, where application integration is widely accessible and where financial institutions, traders, and funds administrators can share data and transactions. When you couple that opportunity with the world's preeminent financial

transaction network, it presents some exciting commercial prospects for the financial industry. "SWIFT provides the global messaging platform that supports transactions between the world's major financial institutions," commented Allan Spalding, Regional Manager, and SWIFT Partner Solutions. "The advent of new technologies such as Web services, which make the integration of applications with SWIFT faster, offers additional opportunities for our institutions."[44]

Web services offer a platform neutral approach for integrating STP applications, so that it can be used to integrate diverse systems in a way supported by standards rather than proprietary systems. The ability of a financial institution to have access to real-time trade related information spanning across multiple companies, in-house departments, applications, platforms, and systems is one of the most important driving factors behind the adoption of Web services. Moreover, Web services provide secure, reliable and cost-effective seamless messaging services. Facing the new evolution of Web services, SWIFT proposes to investigate and analyze to develop a new XML-based message, utilizing this promising technology for its business area.

We are happy to see that SWIFT is migrating to its new IP-base SWIFTNet, the new software using Web services integration principle. As the Web services technology and components adopting in SWIFT community, we believe that more and more corporations may be interested in gaining the benefits of Web services, as this result, SWIFT will be known not only in banking industries but also in wilder areas such as manufacture, corporations and so on.

11 Summary

The technology of Web services has been a popular issue and enthusiastically explored by many in the financial industry. Today, all financial services were required to share information and services, and the challenge is underlying that their different systems for fundamental business processes can not easily be interoperated. Web services in financial services can mitigate a great deal of the business challenges inherent in sharing complex financial data across heterogeneous systems and business processes. Because Web services can offer some potential in distributed systems and significant opportunities of revenue growth and improved efficiency for companies, as leading adopters of information technology, financial institutions are likely to quickly embrace Web services and incorporate Web services into their own technology infrastructure.

In this thesis, we have taken a look at several cases that Web services were applied successfully in financial industry. They presented their use of Web services built upon to reveal solutions that fit individual financial institutions and their specific

needs. In order to use Web services successfully, financial institutions must consider a solution that enables them to leverage existing IT investments and provides them with the capability to manage the Web services. On the other hand, Web services could bring about huge efficiency increases into an IT environment and significantly reduce total cost of ownership.

- Interoperability

As XML Web services one of most significant characters, language and platform independency make application developing simplified that more and more industry are embracing to use Web services integration. Financial industry, as highly regulated data intensive industry, involves many various systems and languages providing a wide variety of services for different customers, clients, and users, and will be the big beneficiary from Web services technology.

- Modular programming and Re-using

Not only Web services promote modular programming that multiple organizations can communicate with the same Web service, also they can provide a standard interface that could be used by a wide range of other applications, both internal and external accessed by any system via intranet or Internet. So far about two-thirds of Web services implementations in financial industry deal with internal integration, rather than integration with customers, partners, or public Web services. With the Web services security technology maturing, financial industry would be able to the biggest service provider for the integrated, secure, and reliable end-to-end transactions.

- Leveraging existing IT investments

Web services enable financial institutions to provide a foundation for integrating business processes, legacy systems, databases and workflows within and across organizations. Web services unified architecture leverages existing infrastructure such as network, legacy systems, database, so financial institutions do not have to adopt entirely new software.

- Cost-efficiency

Web services can significantly reduce the costs of enterprise application integrations (EAI) and business-to-business (B2B) communications, thus offering financial industry tangible returns on their investments. The cost-effective is a key feature of Web services from two aspects. One hand, it decreases implementation complexity and enable to be re-usable, thus reduces the cost and time required for integrating different systems on different platforms. On the other hand, cost

reductions result directly from these operational improvements. The improved integration of applications software that results from Web services results in many manual processes being automated, thus speeding up operations and making more information available to management faster. Eventually, revenue also increases as the use of Web services because Web services allow financial industry provide more data and value-added services to customers and attract new customers and expand into new markets without significantly increasing the company's infrastructure.

Part IV: Web services Implementation-Import/Export International Trading System

In this part, we propose to develop several Web services that could be used in the field of international trading area. These Web services are involved in many companies, financial institutions such as banks and insurance companies, and some other service providers like shipping companies. The main aim of this project is using Web services to move the international trading process into a certain automatic level via Internet, reduce the cost of operation time, paper work, and labor work, meanwhile to cope with the expecting XML-based SWIFT message standards and eventually achieve high-level performance, operating automatically and seamlessly in the process of Letter of Credit.

12 Methodology

We use the Hall's morphology to analyze and develop our Import/Export international trading system. The Hall's morphology is a systematic way and a powerful tool in development and implementation of a system. This systems thinking approach, in contrast, focuses on how the thing being studied interacts with the other constituents of the system. This means that instead of isolating smaller and smaller parts of the system being studied, systems thinking works by expanding its view to take into account larger and larger numbers of interactions as an issue is being studied. The character of systems thinking makes it extremely effective on the most difficult types of problems to solve: those involving complex issues, those that depend a great deal dependence on the past or on the actions of others, and those stemming from ineffective coordination among those involved. For this reason, we apply the Hall's morphology system thinking approach to analyze, design and built our International Trade prototypes.

As system thinking is fast becoming a powerful tool for decision-making and organizational change, the Hall's morphology[45] has been used in this thesis as a system thinking methodology. During the development of systems engineering theory, Hall presented an initial framework that is quite adequate. This framework is a three-dimensional morphological box. Two dimensions of the box are the phase and the steps of systems engineering. This thesis seeks to analysis the real life case associated with three phases of Hall's matrix (Figure 9): the program planning phase, the project planning, and development and design.

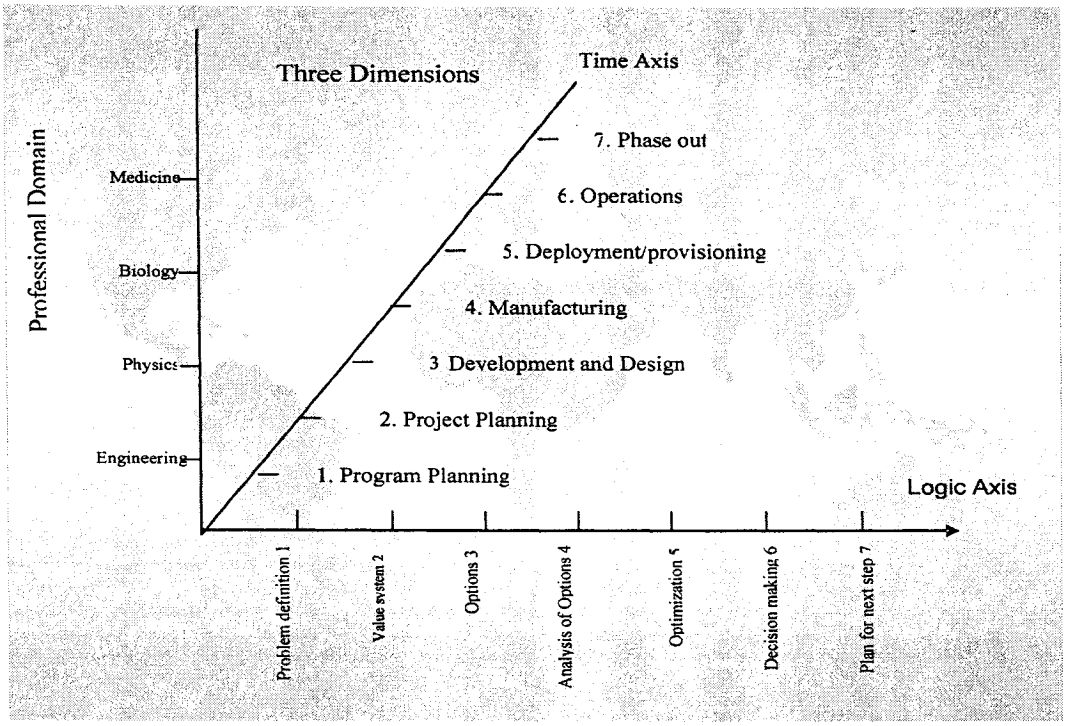


Figure 9: Hall's Systems Engineering Morphology

The aim of this project is to find the best way to integrate all partners of international trade together, implement easily and effectively. Using Hall's morphology system thinking approach, we analyze the problem from two aspects: operation and economy. From operation aspect, the aims of our project are to maximize project lifetime and to minimize developing time, as the result to maximize performance. From economy aspect, the aims are to minimize developing cost and operation cost, finally to achieve the minimum total cost (Figure10). Through conducting case studies before, we know Web services are the best solutions to provide efficiency integrations for complex financial data across heterogeneous systems and business processes, which help us to decide using Web services to implement the international trading system.

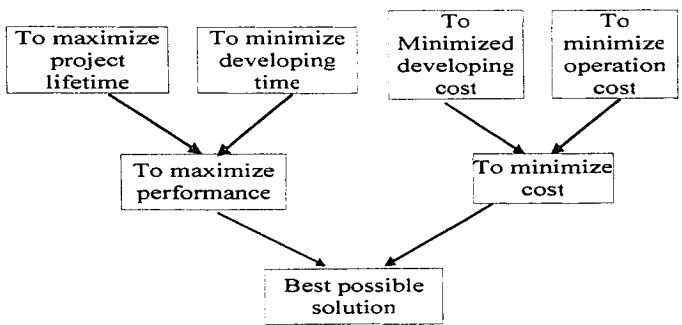


Figure 10: Aims of Projects

13 Program Planning

As we know, the whole process of international trade involves in many companies, financial institutions, and organizations, such as traders, banks, insurance companies, shipping companies, SWIFT and so on. Among these partners, most of data and documents are transferred and shared by mail or manually. With the Internet technologies being used wider and wider, many organizations and companies are embarking to provide their services via Internet. As demonstrated before, SWIFT is starting to provide their services via Internet instead of its proprietary network, and to develop a new XML-based message, which will experience an inter-bank communication evolution in the trade services area and bring a new era for online financial business services. To cope with these renovations in SWIFT, keep competition and create new business chance and challenges, there are requirements for other partners to provide their services via Internet as well.

The Figure11 illustrates the main steps of the international trade. International trade starts from the signed business contract, through Letter of Credit (L/C) issuing and encrypting, involving in shipping and insurance documents, completes until L/C claimed and paid. Many partners, including traders, banks, insurance and shipping companies, participated in international trade process, and all partners are linked by L/C. L/C take critical roles in the entire procedure of international trade, and link all partners together. However, at the present, the tradition way is applying L/C manually, and all application information of L/C is processed by human being, which leads to abundant paper work and labors. Obviously, it is not time-effective, thus we need to find a solution which could automate international trading somewhat in terms of sharing data and transaction information, and be integrated easily and high efficiently as well.

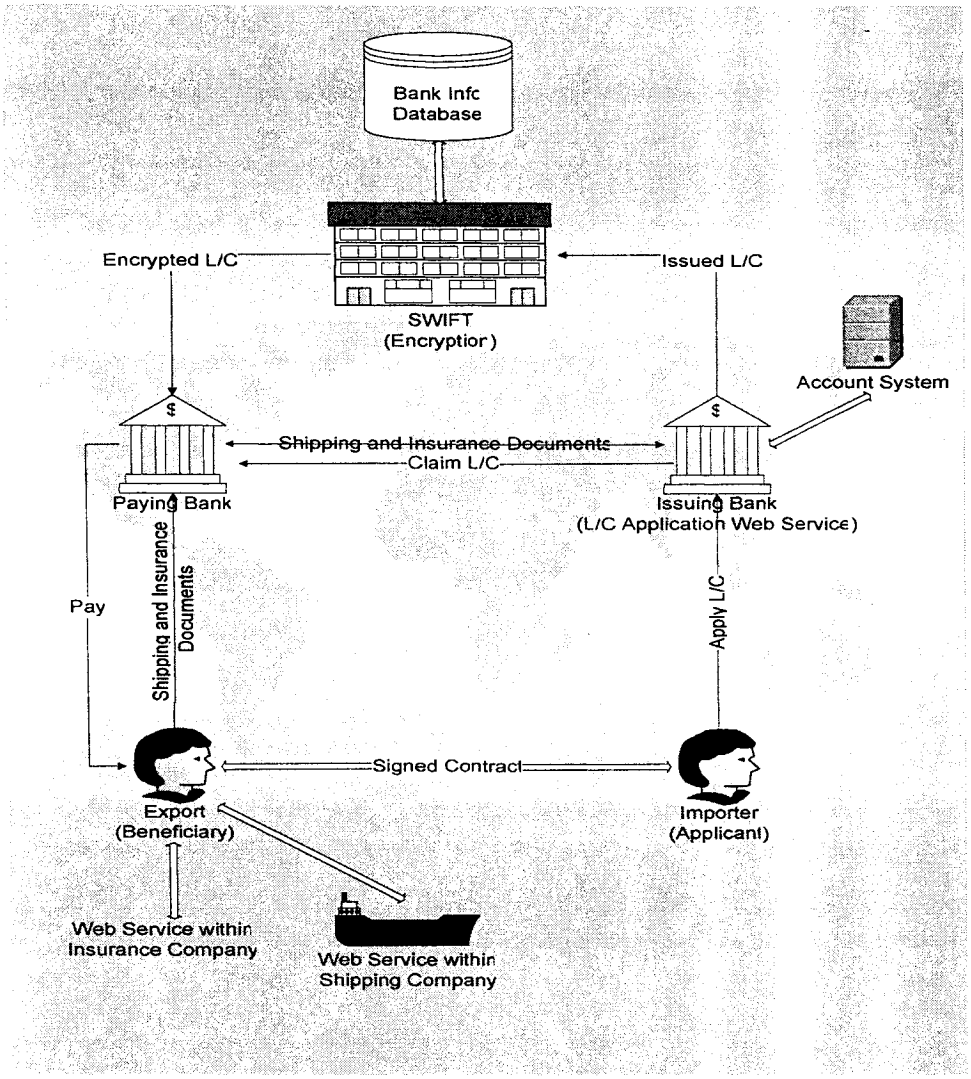


Figure 11: The International Trading System Flowchart

Consequently, in order to understand international trade thoroughly, we are going to introduce the basic concept of Letter of Credit at the first.

13.1 Introduction to Letter of Credit (L/C)

A Letter of Credit (L/C) has long been recognized as a cornerstone of foreign trade for it reduces payment risks when shipping goods to a foreign country. The letter of credit (L/C) is a document issued by a bank at the buyer's request in favor of the seller. Contained within the letter of credit is the issuing banks promise to pay a specific amount of money upon receipt of specific documents. Documents such as shipping date, insurance, arrival in port, terms of sale and any other conditions have been put forth by the two agreeing parties. When a seller receives a letter of credit, he or she has to compare the pro forma invoice to the actual letter of credit to make

careful notes for any discrepancies. Payment will not commence unless all the terms are met. The L/C process and roles are well defined. Figure12 illustrates the process.

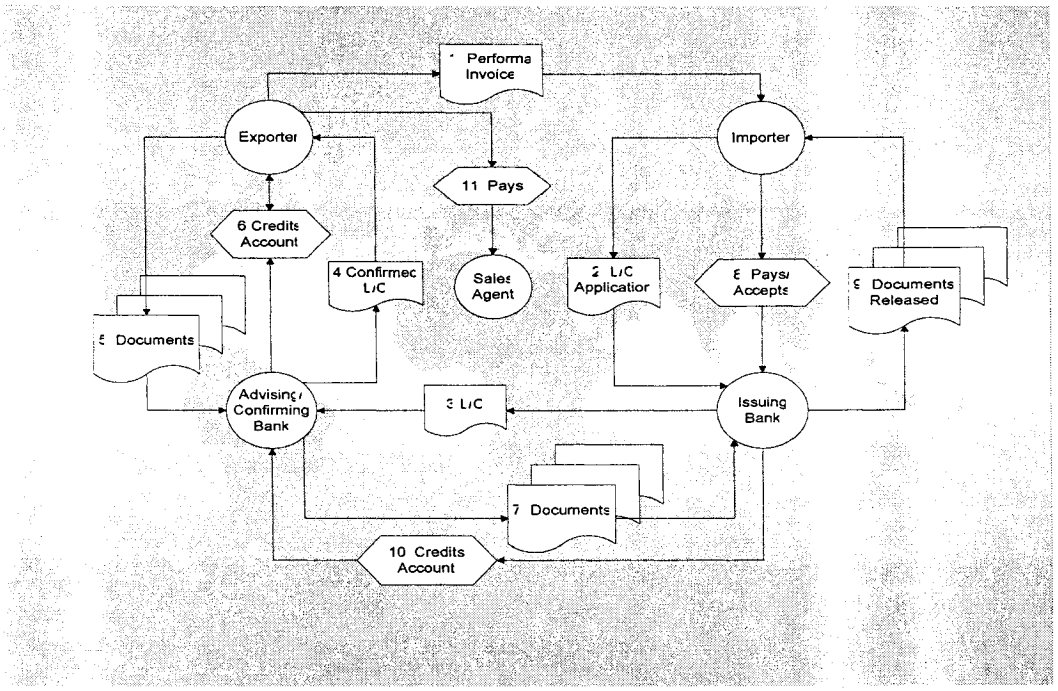


Figure 12: Flow Chart of L/C

1. The exporter sends a Performa invoice containing the terms and conditions of sale to the importer;
2. The importer applies to his bank for a letter of credit (L/C);
3. The issuing bank transmits the L/C to the advising/confirming bank in the importer's country;
4. The advising bank advises the exporter of the arrival of the L/C and may also confirm the L/C;
5. The exporter checks the L/C and presents the required documents to the bank;
6. If the documents have no discrepancies, the advising/confirming bank credits the exporter's account;
7. The advising/confirming bank sends the documents to the issuing bank;
8. The issuing bank releases the documents on payment or;
9. Acceptance of a bill of exchange. Bill of exchange is a non-interest bearing written order used primarily in international trade, binding one party to pay a fixed sum of money to another party at a predetermined future date. Bills of exchange are similar to checks and promissory notes. They can be drawn by individuals or banks and are generally transferable by endorsements. If these bills are issued by a bank, they can be referred to as bank drafts. If they are issued by individuals, they can be referred to as trade drafts.

10. The issuing bank credits the account of the advising/confirming bank.
11. The exporter pays his sales agent.

13.2 Challenge for L/C

Although Letter of Credit is such an important document that takes the significant role during the whole trading process, there are still a number of challenges associated with the L/C process:

- The process is manually intensive, time consuming, and prone to error. Documents are typically mailed or faxed. Some banks support the download of documentation but only for subsequent printing. The information must then be re-keyed.
- Some banks have introduced PC based software to automate a portion of the process but the software is bank specific and must be loaded onto the customers' PC platforms.
- There is not a standard for a L/C or supporting documents. Some financial institutions use the SWIFT format for the L/C. (Note: SWIFT is an acronym for Society for Worldwide Inter-bank Financial Telecommunications. It is an international electronic inter-bank messaging system.) Documents such as the bill of landing may be sent via an EDI format.
- Although the process is well defined, automation has been slow in coming for predominantly one reason- proprietary interfaces inhibiting the exchange of information. An importer or exporter may deal with a variety of banks. It is impractical for the importer/exporter to install and maintain bank-specific software. Likewise, a bank may deal with many importers and exporters. It would be an impossible task to quickly implement and maintain unique interfaces for every business-to-business relationship.

L/C is critical for international trade, it is a linkage with all related partners, and all information inside is shared and used by each of them. SWIFT as the governance of L/C is going to provide XML-based massage standards and posting their service via Internet instead of its old MT message format via its proprietary web. As a result, it is urgent to ask for other partners to provide their services online or build very convenient services via Internet.

13.3 Insurance and Shipping Issues Related in International Trading

Similarly, an exporter may also deal with many insurance companies and shipping companies. For each business, exporter is asked by importer to provide proper insurance and shipping means based on their contract.

In insurance company, there are three relationships existing in their business, insurance company and agent, agent and customers, and insurance company and customers. The insurance company offers a product range, while the agent offers a selling and distribution channel. However, the increased competition between insurance companies will decrease the chances to benefit. The relationship between the agent and customer is similar to the one between the customer and the insurance company. There is no real difference whether customers buy an insurance policy from agents or from insurance company, where agent is just a linkage between customers and insurance company. The only thing she or he cares about is price. Aim to independent less on the agent, and increase and extend its business for more customers, insurance company is asked to provide more convenient services for both agent and customers.

Shipping company is facing the same situation as well. Insurance companies and shipping companies also can simplify the process of applications if they could provide similar services for applications like banks can do for L/C applications. Therefore, for insurance and shipping companies, it is also significant to find a solution over Internet to simplify applications and extend their business and market.

14 Project Planning

We think of Web services are the perfect solutions to achieve automation in international trade process via Internet because they provide efficiency integrations for complex financial data across heterogeneous systems and business processes. Since all corporations in the procedure of international trade have different legacy systems, platforms and languages, Web services are wonderful tools to handle and integrate them together. Without thinking about SWIFT, we divide the rest parts of the international trade system into three sub systems: L/C Application system, Shipment Quote system, and Insurance Quote system, and simulate three implementation prototypes of Web services between traders and banks, shipping company, and insurance company without involving any security issues.

As the Figure13 shown, these three prototypes are designed to enable the bank, shipping company and insurance company to link among its corporate client's ERP system and accessed via Internet without making major modifications to their legacy systems. Furthermore, they also have the following distinctive features:

- Flexible for using different systems in the combination environment of Internet and propriety network
- Easy to keep tracking the process of international business management
- To reduce the traditional paper work

- To integrate all related systems and reduce time delay
- To achieve cost-efficiency

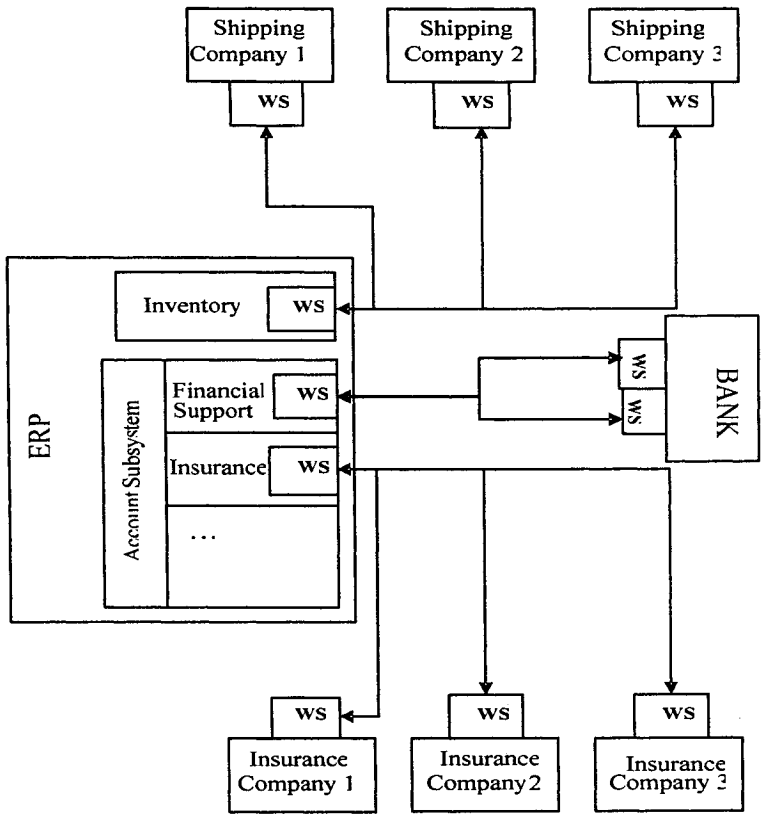


Figure13: The Profile of the International Trade Project

14.1 The Letter of Credit Prototype

We designed the L/C prototype on the basis of ERP system. This module was built under the account subsystem of ERP, and allows users to apply L/C automatically. Once a company has a good relationship with a particular bank, they usually do all business with it because of convenience and reliabilities except that there are some other reasons or situations to change their business bank. For this reason, the module we built was deployed Web services from the bank that has a good relationship with the exporter. We designed two Web services, AppLC Web service and Issue Web service for the bank to deal with L/C processing. Meanwhile, we suppose that the corporation has an ERP system, so we developed Web service Interface under ERP environment rather external to it, and these Web services execute data transaction with external Web services from banks. Figure 14 is the flowchart of this prototype.

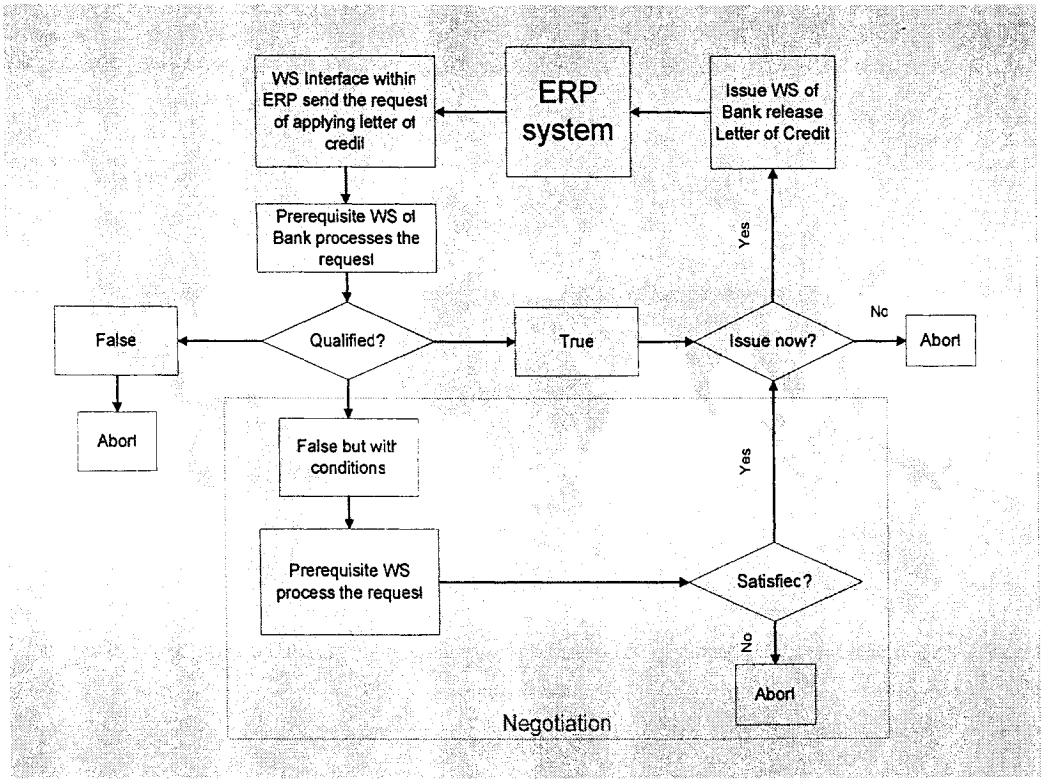


Figure 14: L/C Application Web service Flowchart

L/C was previously exchanged between businesses and their banks on hard copy. With the new system, data on L/C will be in electronic format and messages communicate on the XML-base. Computers of businesses will be linked with the bank's, speeding up the administrative process. The L/C application Web service interface we built takes charge of the role of sending the request of applying and issuing L/C. In addition, banks usually check clients' credit rate and the balance in their account before issuing a L/C, and we address negotiation in this Web service application as well. This negotiation depends on if the credit rate of the applicant matches a certain account balance; that is to say, systems decide what deposit they required when they received applicant's request. For example, when company ABC apply to his bank for a L/C with the amount US$50000.00, by searching ABC's credit rate in bank's database which is 8, depending on the bank's regulation, ABC was asked by the bank to deposit the 10% of the L/C's into his account. That is to say, 50000*10%=5000 has to be held as the guarantee payment. If their account has enough balance, their L/C could be issued immediately. Otherwise, system will respond a message which explains the minimum guarantee payment that bank required, or with conditions that L/C can be issued with a higher interest rate. Then, company ABC is going to decide whether they need this letter of credit right away or choose another bank. If they want to continue business with the bank, and they approve to accept these conditions, and systems are able to issue their L/C too. This

Web service makes bank directly exchange detailed data on L/C with its corporate clients online, and achieve e-negotiation in a certain level as well.

Both banks and traders will enjoy increased convenience provided by Web service, such as transaction information communications, credit rate searching and account balance confirmation. L/C is processed automatically instead of traditionally manually, and all information and data are based on more manageable electronic format instead of paper work, all of which enable the entire L/C application process to achieve the intensive time and cost efficiency. After L/C has been issued by a bank, the XML format L/C could be shared with those applications Web services provided by SWIFT, thus dramatically increase efficiency in international trade process.

14.2 Shipping Web services Prototype

Web services make a renovation for shipping company. Customers can do everything over Internet under the help of Web services, including calculating fees, choosing pickup and delivery locations, filling all kinds of international documents and so on. It benefits both customers and shipping company for cost and time effective. Shipping Web service Prototype is designed between traders and shipping companies. We suppose the exporter also has an ERP system, thus we need to build a Web service interface within ERP system which links ERP system and external Web services from shipping companies. Meanwhile, we build three Web services from different shipping companies posted in their web site, which offer new distributions channel for those shipping companies, and offer agents more flexibility in offering products and services from the available product ranges as well. Figure15 shows the framework of this prototype.

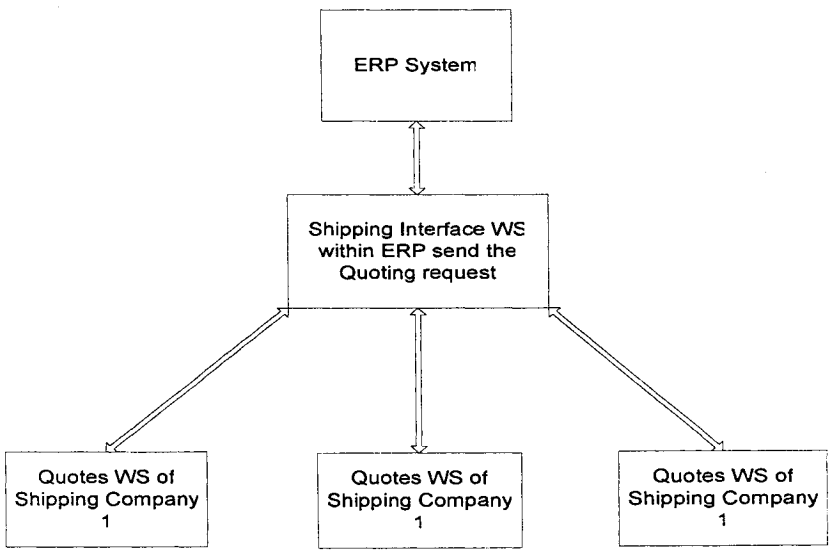


Figure15: Shipping Quote Web service Flowchart

When an order has been confirmed, Web service Interface within ERP is able to search orders from its database, and thus company need to find a proper shipping company to delivery their produce. Then, under the help of Web service Interface deployed within ERP system, the prototype will look for relevant Web services from shipping companies, and send its request of shipping fee quoting automatically. As soon as those shipping companies' Web services get the request, they will reply their quoting calculation response at once. Immediately, Web service Interface in ERP system received these responses, and parsed all information to decision-makers. During this entire process of sending request and exhibiting response, no human being involve in, it is totally automatic.

14.3 Insurance Web services Prototype

Insurance policy is also very important documents in international trading. It provides protection for goods shipping, and reduces risks for both importer and exporter. Due to this, all requirements of insurance are listed under importer's request when the contract was signed, and usually written under the conditions or terms of L/C. For exporter, they need to find a proper insurance policy which both satisfy his L/C's conditions and decrease the cost as much as possible. The traditional way is looking for one policy under agents broadcasting or by exporter-self. However, it is relative limited and slow. Web services make it relatively easy for customers or agents to switch from one insurance company to another. Besides, Web services can affect the behavior of the customer. For instance, when a customer shops around for the right insurance policy, the new possibilities offered by Web services serve to present lots of information to her or him. Based on this information, the customer can pick the policy that suits her or his needs best, fast and convenient. To those customers who only go to one insurance agent, they will be benefited by a wider choice from insurance agent with the use of Web services technology.

As a consequence, we build the Insurance Web services Prototype to present the feasibility of Web services application for insurance company. This is a similar Web service quote calculator as the shipping company did. It includes a Web service Interface within ERP system and Web services from insurance companies. Its frame looks like Figure16.

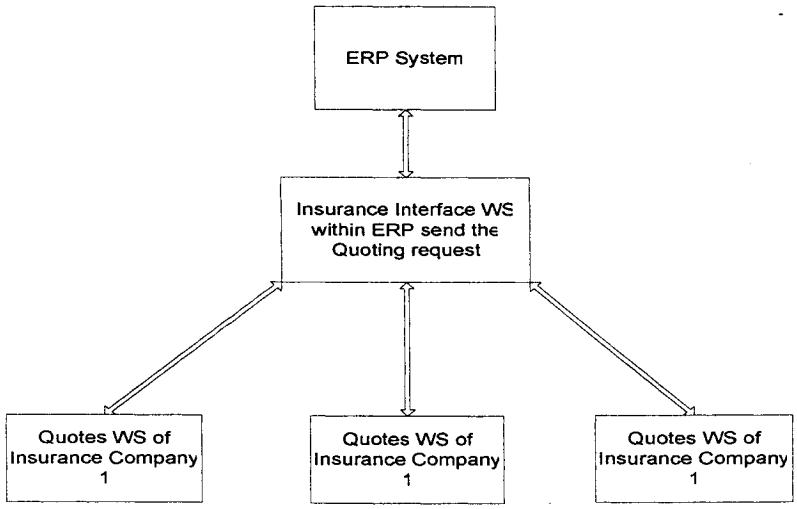


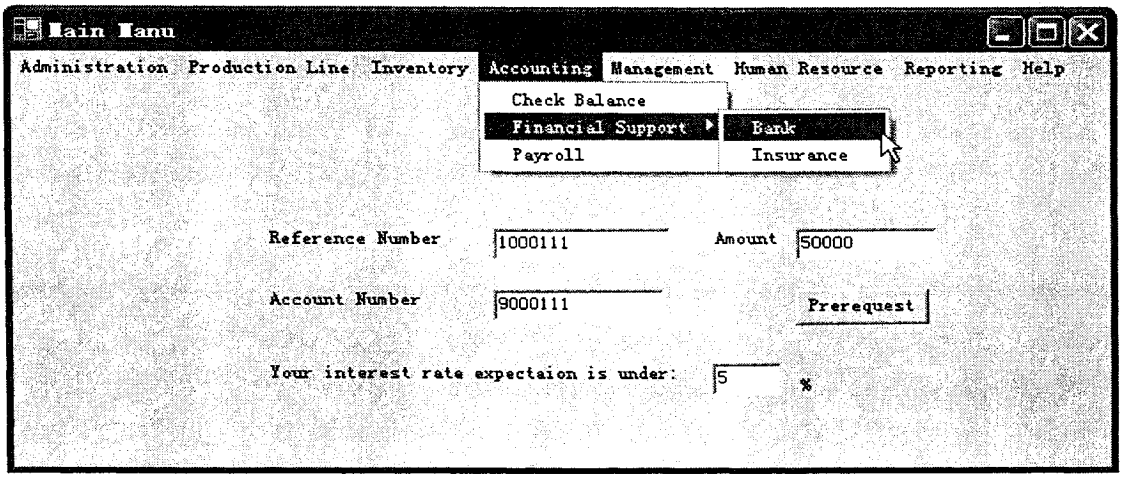
Figure 16: Insurance Quote Web service Flowchart

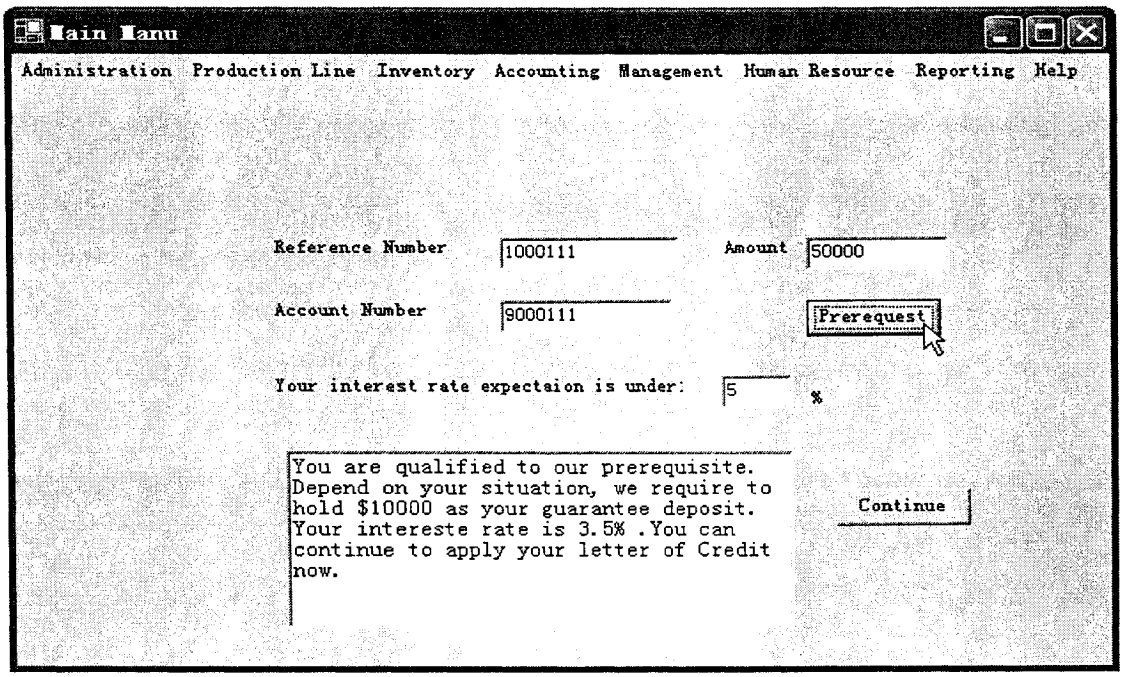
15 Development and Design

At the present, there are many technology tools to develop a WS application, such as Microsoft .Net Framework, IBM WebSphere SDK for Web services (WSDK) that is based on Java technology, etc, and each developing tool has its own features and benefits. We choose the Microsoft Visual Basic .NET to build Web services, because it can provide a collection of classes and tools to aid in development and consumption of XML Web services applications. The .NET Framework is also built on standards such as SOAP, XML, and WSDL to promote interoperability with non-Microsoft solutions. In addition, no matter in banking services, insurance services or shipping company, they all need to process huge data transactions which require sound databases systems, here we choose SQL server 2000 as our database environment, and using ADO.net technology to connect and communicate data.

15.1 Letter of Credit Application Prototype

On the basis that the trader has an ERP environment, we built L/C application module under Account sub-system of ERP. As soon as user types their request of L/C application information, the Web service is called by clicking prerequisite button. Shown as Figure 17, the Web service passes all those information to bank, and executes checking applicant's credit and account balance from its database.


Main Menu
Administration
Production Line
Inventory
Accounting
Management
Human Resource
Reporting
Help
Check Balance
Financial Support
Payroll
Bank
Insurance
Reference Number
1000111
Amount
50000
Account Number
9000111
Prerequest
Your interest rate expectaion is under:
5
%


Main Menu
Administration
Production Line
Inventory
Accounting
Management
Human Resource
Reporting
Help
Reference Number
1000111
Amount
50000
Account Number
9000111
Prerequest
Your interest rate expectaion is under:
5
%
You are qualified to our prerequisite. Depend on your situation, we require to hold $10000 as your guarantee deposit. Your intereste rate is 3.5% .You can continue to apply your letter of Credit now.
Continue

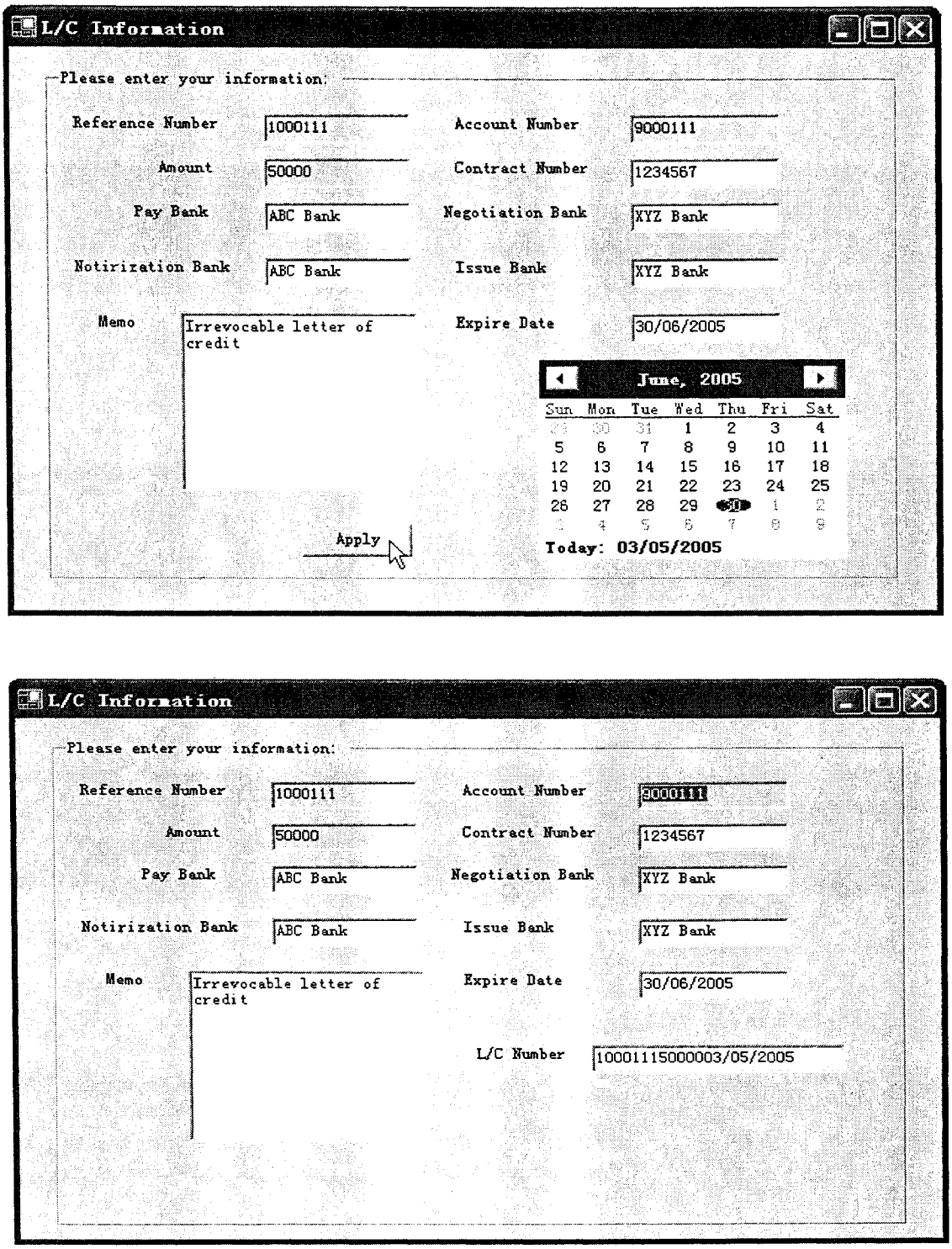


Figure17: L/C Application Prototype

After completing its checking, it gives the response about qualified or not. If qualified, prototype will allow user to continue entering its L/C information and wait for bank issuing the. Otherwise, Web service returns a message to explain the reason why the applications was rejected, or gives conditions that bank requires. When Web service interface in ERP system receives the response, it will execute its negotiation

with bank's prerequisite Web services based on tolerate conditions that the applicant has set such as interest rate. Depend on the result of negotiation, system will call issue Web service or abort the entire procedure. If conditions are approved by ERP system, the Issue Web service will accept all data from the applicant and double check the accuracy, then insert new record into its database for L/C information and reply the L/C number to client. As soon as all process completed, XML-based electronic LC message will be generated and can be used for other partners in international trading. The entire procedure are involved in human being operation, because financial decision making is very complicated, and decision makers need to set many regulation or limitation for their negotiation with bank. Next Figure18 and Figure19 depict the WSDL files of AppLC Web service and Issue Web service.

```
<?xml version='1.0' encoding='utf-8' ?>
- <definitions xmlns:http='http://schemas.xmlsoap.org/wsdl/http/' xmlns:soap='http://
schemas.xmlsoap.org/wsdl/soap/' xmlns:s='http://www.w3.org/2001/XMLSchema' xmlns:s0='http://
tempuri.org/' xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/' xmlns:tm='http://
microsoft.com/wsdl/mime/textMatching/' xmlns:mime='http://schemas.xmlsoap.org/wsdl/mime/'
targetNamespace='http://tempuri.org/' xmlns='http://schemas.xmlsoap.org/wsdl/'>
- <types>
- <s:schema elementFormDefault='qualified' targetNamespace='http://tempuri.org/'>
- <s:element name='ApplyLC'>
- <s:complexType>
- <s:sequence>
<s:element minOccurs='0' maxOccurs='1' name='RefNo' type='s:string' />
<s:element minOccurs='1' maxOccurs='1' name='Amount' type='s:short' />
<s:element minOccurs='0' maxOccurs='1' name='AcctNo' type='s:string' />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name='ApplyLCResponse'>
- <s:complexType>
- <s:sequence>
<s:element minOccurs='1' maxOccurs='1' name='ApplyLCResult' type='s0:Result' />
</s:sequence>
</s:complexType>
</s:element>
- <s:complexType name='Result'>
- <s:sequence>
<s:element minOccurs='0' maxOccurs='1' name='str' type='s:string' />
<s:element minOccurs='1' maxOccurs='1' name='boo' type='s:boolean' />
</s:sequence>
</s:complexType>
</s:schema>
</types>
- <message name='ApplyLCSoapIn'>
<part name='parameters' element='s0:ApplyLC' />
</message>
- <message name='ApplyLCSoapOut'>
<part name='parameters' element='s0:ApplyLCResponse' />
</message>
- <portType name='ApplySoap'>
- <operation name='ApplyLC'>
<documentation>Method to apply a letter of credit.</documentation>
<input message='s0:ApplyLCSoapIn' />
<output message='s0:ApplyLCSoapOut' />
</operation>
</portType>
- <binding name='ApplySoap' type='s0:ApplySoap'>
<soap:binding transport='http://schemas.xmlsoap.org/soap/http' style='document' />
- <operation name='ApplyLC'>
<soap:operation soapAction='http://tempuri.org/ApplyLC' style='document' />
- <input>
<soap:body use='literal' />
</input>
- <output>
<soap:body use='literal' />
</output>
</operation>
</binding>
- <service name='Apply'>
- <port name='ApplySoap' binding='s0:ApplySoap'>
<soap:address location='http://localhost/LCApp/ApplyLC.asmx' />
</port>
</service>
</definitions>
```

Figure 18: ApplyLC Web service WSDL file

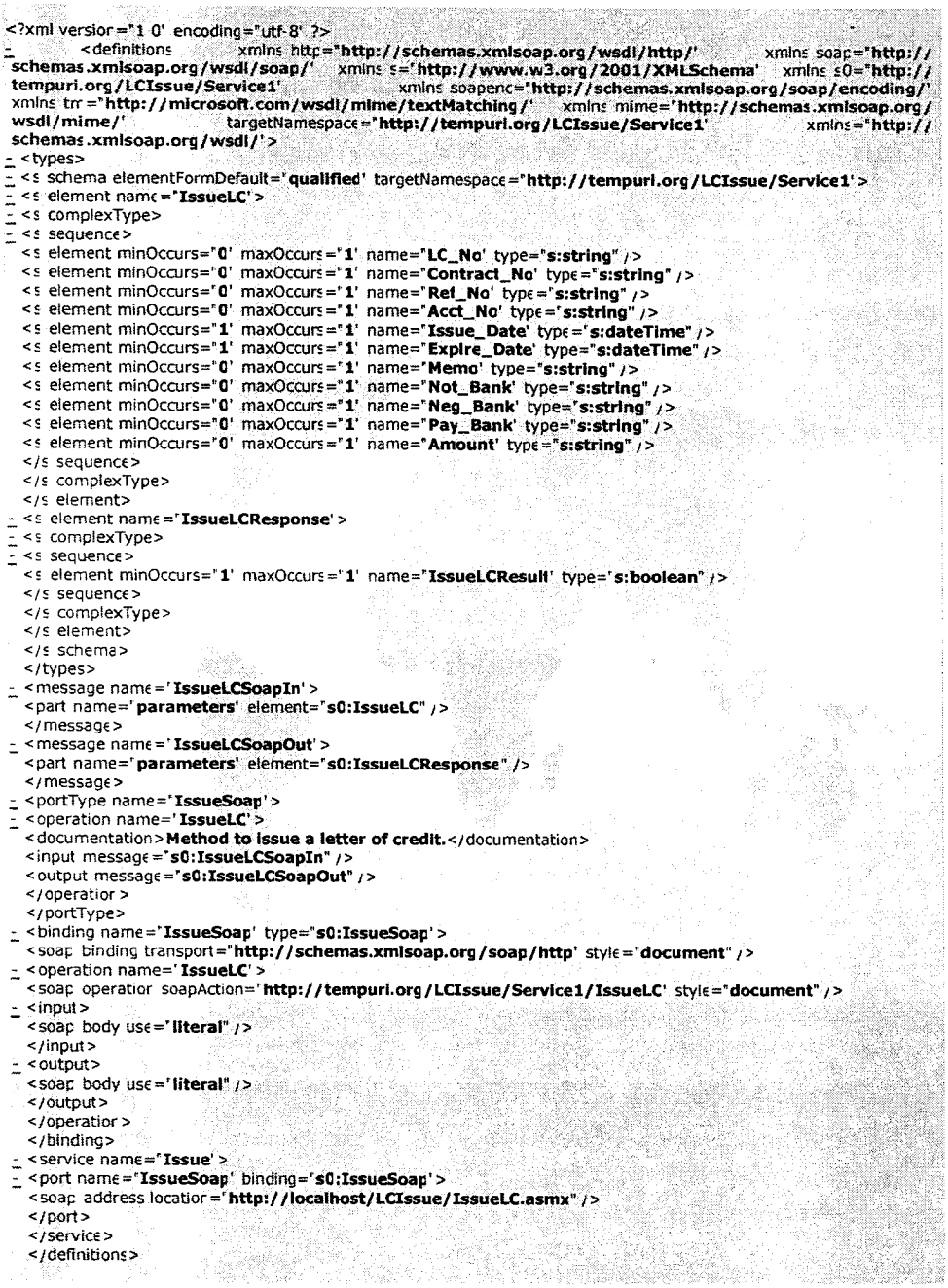


Figure19: Issue Web service's WSDL file

15.2 Shipping Web services Prototype

The Figure20 illustrates the Shipping Web service Prototype. The prototype was built under the module of Inventory in ERP system. When user enters into this submenu and chooses the shipping choice, the prototype reads all confirmed order information directly from its database.

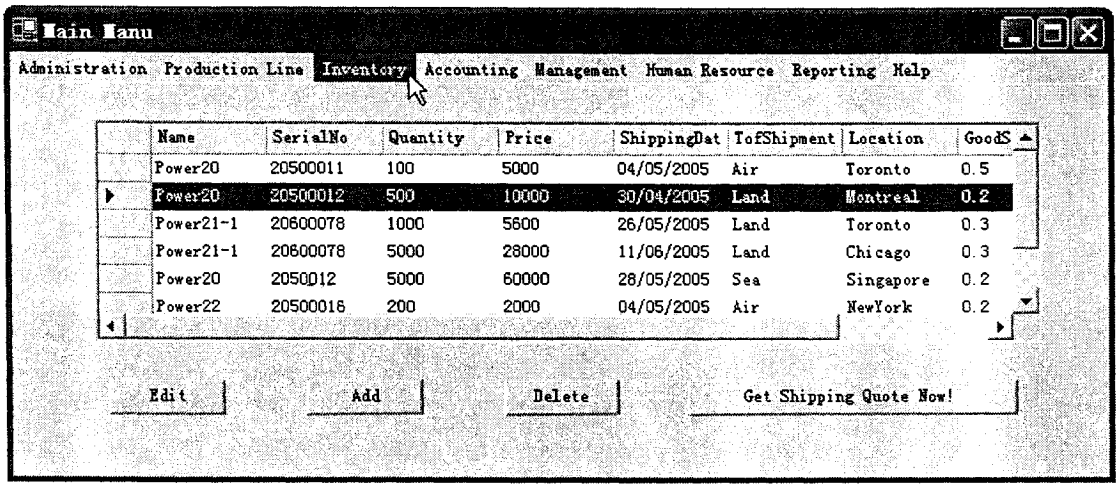


Figure20: The Shipping Quote Module

Once user chooses an order and clicks the button "Get Shipping Quote Now!" the prototype calls the Shipping Web service Interface built under this window. This Web service was linked to other Web services that provided and posted by shipping companies via Internet. After click the quote button, the Shipping Interface Web service call those Web services from shipping companies to execute shipping quoting. Then, the prototype displays the quotes result as soon as it gets responses from those Web services, shown as Figure21.

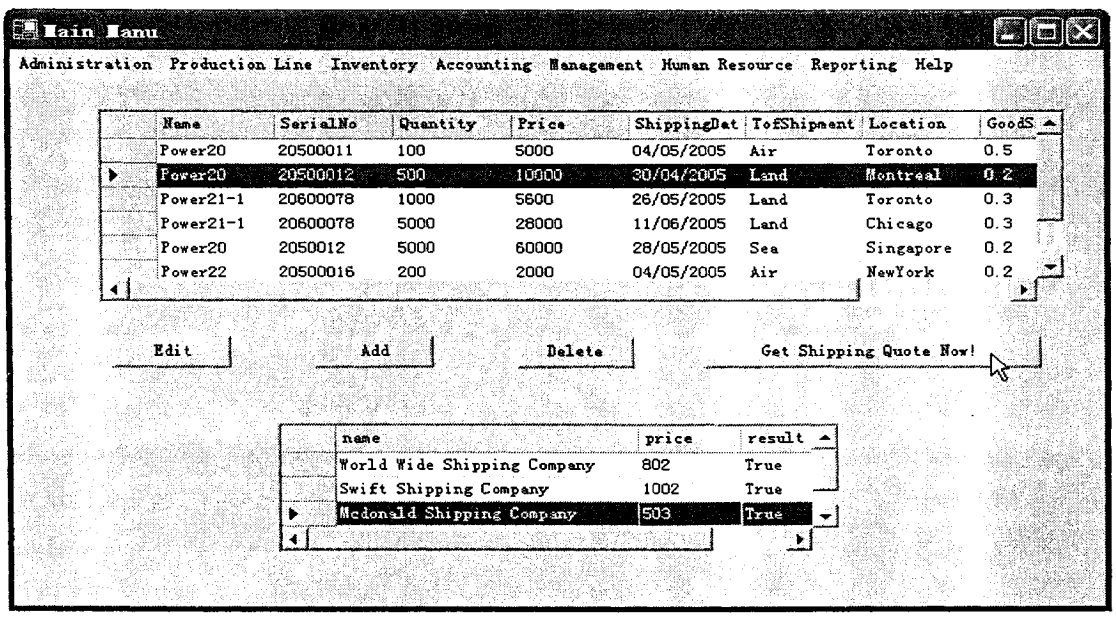
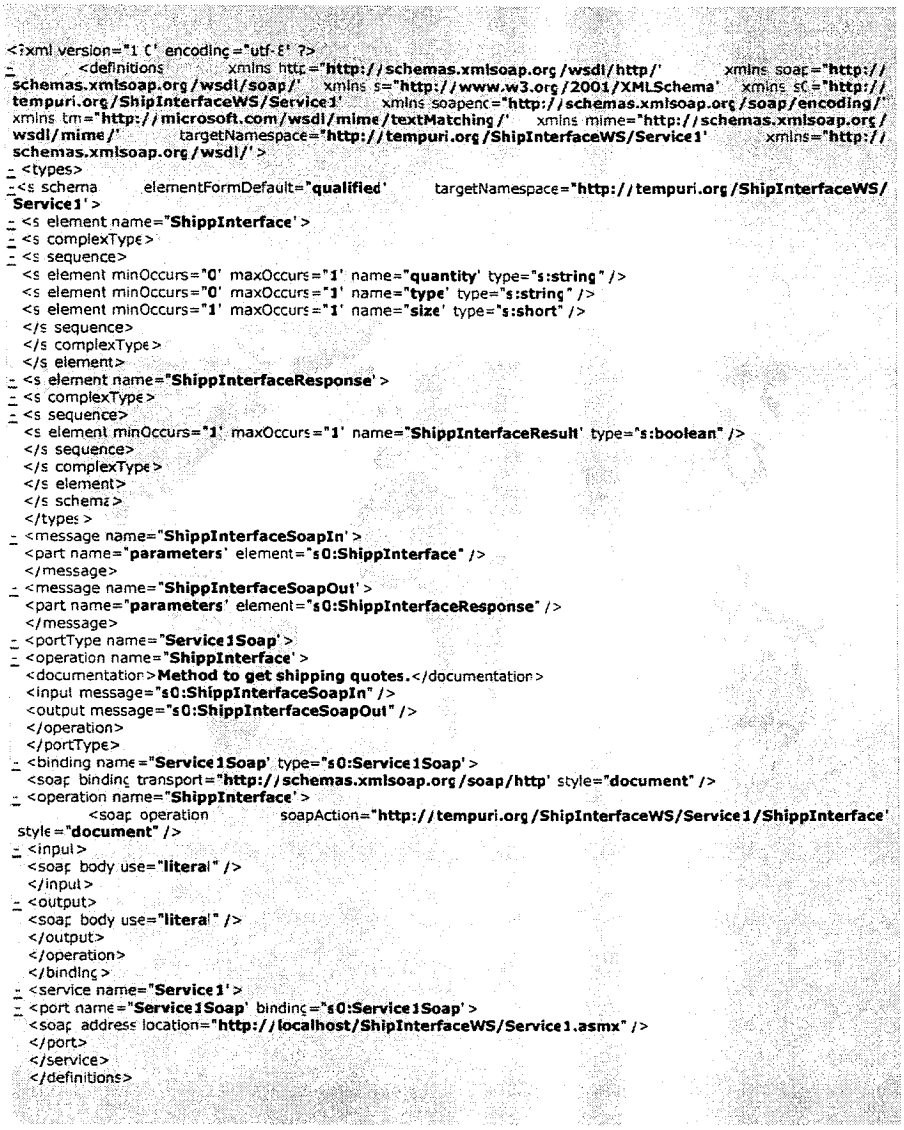


Figure21: The Response of Web services from Shipping Companies

This procedure was triggered by choosing a confirmed order, and the rest steps are ran automatically without human being involved. The messages transferred within the entire procedure are soap messages. The following Figure22 is the WSDL file of the Shipping Interface Web service.



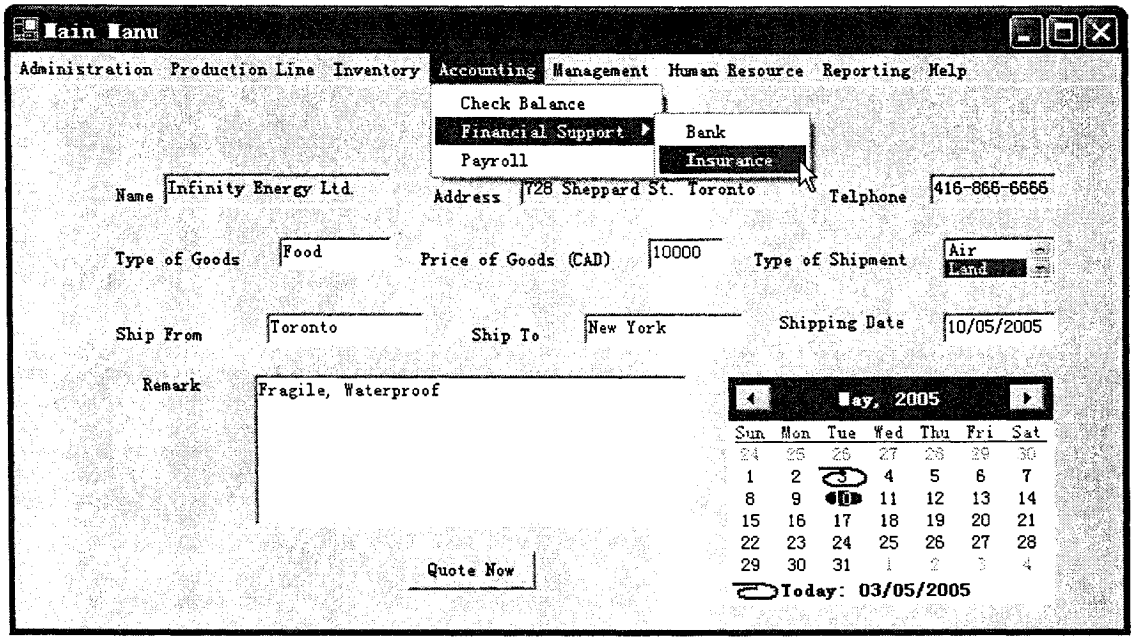
```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://
schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://
tempuri.org/ShipInterfaceWS/Service1" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:mime="http://schemas.xmlsoap.org/
wsdl/mime/" targetNamespace="http://tempuri.org/ShipInterfaceWS/Service1" xmlns="http://
schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/ShipInterfaceWS/
Service1">
- <s:element name="ShippInterface">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="quantity" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="type" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="size" type="s:short" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="ShippInterfaceResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="ShippInterfaceResult" type="s:boolean" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </types>
- <message name="ShippInterfaceSoapIn">
  <part name="parameters" element="s0:ShippInterface" />
  </message>
- <message name="ShippInterfaceSoapOut">
  <part name="parameters" element="s0:ShippInterfaceResponse" />
  </message>
- <portType name="Service1Soap">
- <operation name="ShippInterface">
  <documentation>Method to get shipping quotes.</documentation>
  <input message="s0:ShippInterfaceSoapIn" />
  <output message="s0:ShippInterfaceSoapOut" />
  </operation>
  </portType>
- <binding name="Service1Soap" type="s0:Service1Soap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="ShippInterface">
    <soap:operation soapAction="http://tempuri.org/ShipInterfaceWS/Service1/ShippInterface"
style="document" />
- <input>
  <soap:body use="literal" />
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Service1">
- <port name="Service1Soap" binding="s0:Service1Soap">
  <soap:address location="http://localhost/ShipInterfaceWS/Service1.asmx" />
  </port>
  </service>
  </definitions>
```

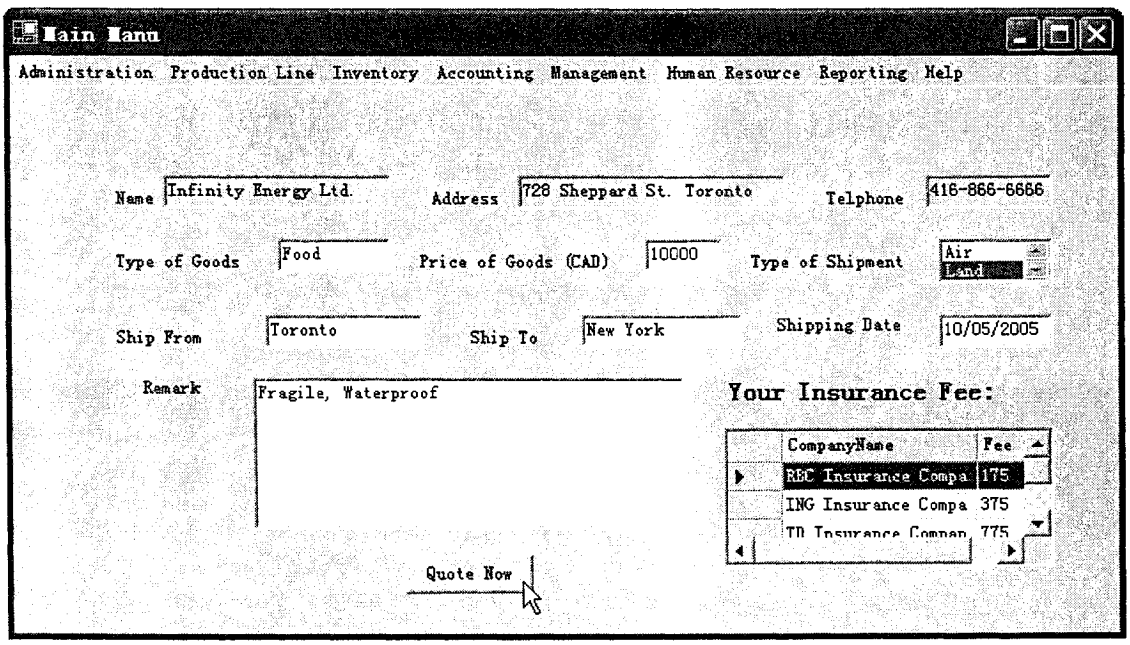
Figure22: WSDL file Of Shipping Interface

15.3 Insurance Web services Prototype

Similarly, the Insurance Web service prototype was designed same as shipping Web services prototype. Its functions are to send insurance quote request and receive responses from three insurance companies. Figure23 presents the shipping Web service and the visual basic code file behind for windows through which users can type all related information about his insurance. This prototype allows users to

calculate his insurance fee on the basis of its price, shipment type, date, from location and to location, and so on.





```
Public Class WebForm
  Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

  This call is required by the Web Form Designer
  <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

  End Sub
  Protected WithEvents Panel1 As System.Web.UI.WebControls.Panel
  Protected WithEvents Label7 As System.Web.UI.WebControls.Label
  Protected WithEvents Label9 As System.Web.UI.WebControls.Label
  Protected WithEvents Label10 As System.Web.UI.WebControls.Label
  Protected WithEvents TelBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents TofGBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents PofGBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents TofGLabe As System.Web.UI.WebControls.Label
  Protected WithEvents PofGLabe As System.Web.UI.WebControls.Label
  Protected WithEvents NameBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents NameLabel As System.Web.UI.WebControls.Label
  Protected WithEvents TelLabe As System.Web.UI.WebControls.Label
  Protected WithEvents RemarkBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents ShipFromLabel As System.Web.UI.WebControls.Label
  Protected WithEvents ShipFromBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents ShipToBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents DofSCalendar As System.Web.UI.WebControls.Calendar
  Protected WithEvents TofSList As System.Web.UI.WebControls.DropDownList
  Protected WithEvents AddrBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents AddrLabel As System.Web.UI.WebControls.Label
  Protected WithEvents QuoteButton As System.Web.UI.WebControls.Button
  Protected WithEvents TofSLabel As System.Web.UI.WebControls.Label
  Protected WithEvents RemarkLabe As System.Web.UI.WebControls.Label
  Protected WithEvents DofSlabel As System.Web.UI.WebControls.Label
  Protected WithEvents ResultBox As System.Web.UI.WebControls.TextBox
  Protected WithEvents DateBox As System.Web.UI.WebControls.TextBox
        Private Agent As New localhost.Quotes
  NOTE: The following placeholder declaration is required by the Web Form Designer.
  Do not delete or move it.
  Private designerPlaceholderDeclaration As System.Object

  Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Init
    CODEGEN: This method call is required by the Web Form Designer
    Do not modify it using the code editor.
    InitializeComponent()
  End Sub

#End Region
  Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    Put user code to initialize the page here
    If IsPostBack Then

      ResultBox.Visible = True
    End If
  End Sub

  Private Sub DofSCalendar_SelectionChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
DofSCalendar.SelectionChanged
    DateBox.Text = DofSCalendar.SelectedDate
  End Sub

  Private Sub QuoteButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles QuoteButton.Click
    Dim s As String
    Dim Fee As String

    s = "Your insurance quote is: " & "$" & Agent.Quotes(TofSList.SelectedItem.ToString, PofGBox.Text).ToString
    Fee = Agent.Quotes(TofSList.SelectedItem.ToString, PofGBox.Text)
    ResultBox.Text = s
  End Sub
End Class
```

Figure23: Windows and VB code

When user clicks Quote, the button event handler executes. The event handler calls the Web service method and passes it the information. Instantly, the application displays a message about the calculating results of insurance fee from three insurance companies. Its SOAP file and WSDL file of Insurance Web service were shown as Figure24.

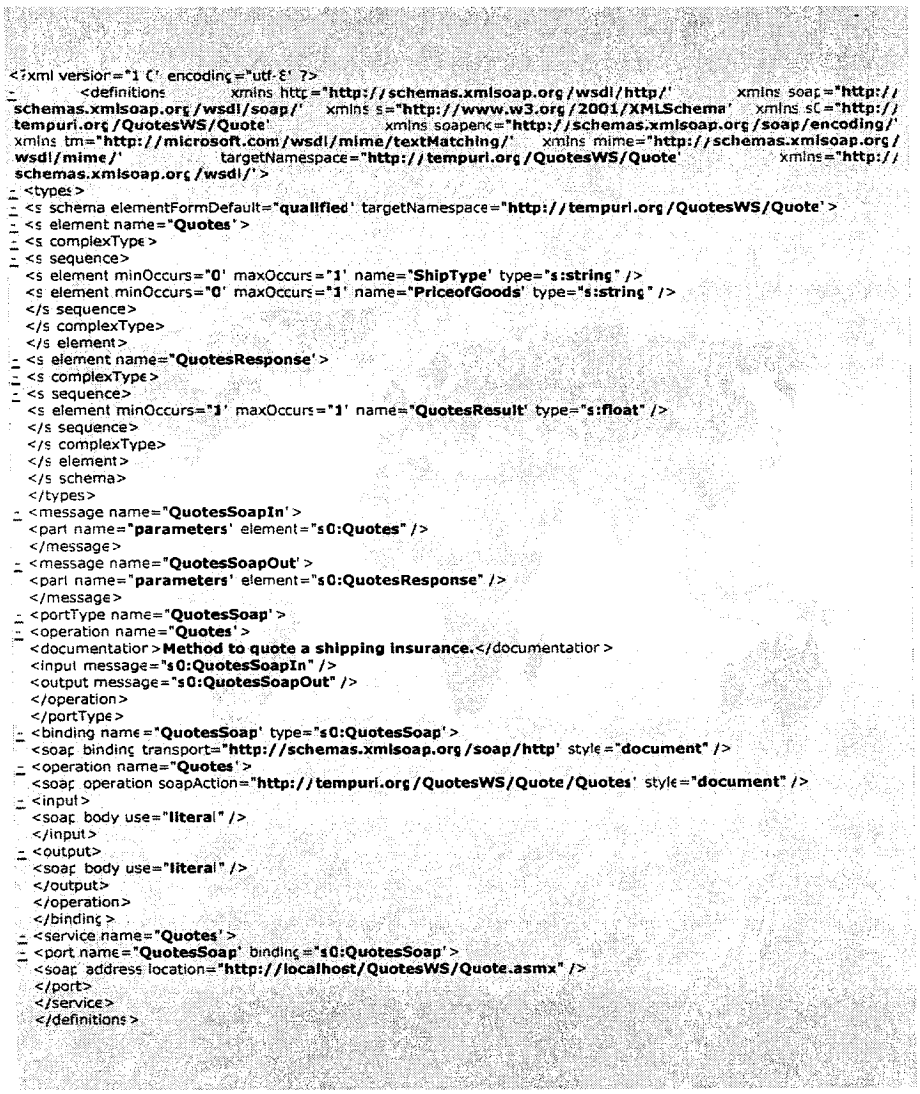


Figure24: Soap Schema of Quotes Web services

Here we only lists several examples of Soap and WSDL files of Web services we developed, all related files, including Visual Basic codes of the project, SOAP and WSDL files of each Web service, are attached in **Appendix** located at the end of this document.

16 Performance

The performance of a Web service is measured in terms of throughput, latency, execution time, and transaction time. Throughput represents the number of Web services requests served in a given time period. Latency is the round-trip time between sending a request and receiving the response. Execution time is the time

taken by a Web service to process its sequence of activities. And transaction time represents the period of time that passes while the Web service is completing one complete transaction. Higher throughput, lower latency, lower execution and faster transaction times represent good performing Web services. [46]

The overall performance of Web services depends on application logic, network, and most importantly on underlying messaging and transport protocols, such as SOAP and HTTP. We are unable to measure of our Web services implementation which based on personal computers. However, with case studies conducted before and these practices for the design and implementation of Web services, we can think of Web services as a perfect replacement for traditional integration problems that are being faced by the enterprises today.

17 Conclusion

By implementing an International Import/Export trade system using Web services, financial institutes will broaden their roles in international trade process. Rather than simply providing a "post office" function for receiving Letters of Credit, notifying the beneficiary, and passing payment, financial institutes provide capabilities to automate the L/C process. Rather than passing documents through the mail and re-keying data, L/C data can be passed electronically, in the appropriate format, to be received directly by the target systems. Moreover, because its format can be flexibly changed by XML technology, the system can offer services that are easily adaptable to changing market conditions, which helps to attract new export customers and improves financial institutes' business agility. Other process participants such as the shipping and insurance companies will also benefit by using Web services as it reduces the cost, time, and errors associated with the international trade process.

At the same time, traders benefit by reduced warehouse costs and better customer relationships due to prompt delivery of goods. Web services solutions also enable exporters to pre-book less expensive shipping methods and insurance policies, because they now know precisely when the goods will be ready to leave the warehouse. There is no doubt that the use of Web services and XML will improve the efficiency and speed of delivery of various banking functions, including corporate financing, the bank's main business.

As has been demonstrated, the Web services used for the international trading system are technologies for easily linking applications on the Internet and are considered the core technology for next-generation e-business. Use of Web services will allow businesses to link any kind of systems more easily than before, making

services attractive as they can be developed quickly and efficiently, and increase capability to explore new products and services that meet customer needs.

Through the analysis and development of Web services drove in this thesis, we have established three WS application prototypes in international trading area, including financial industry, insurance company and shipping company. This International Import/Export Trading System is based on the analysis of interoperation between SWIFT and banking. In addition, it also focuses on some other organizations excluded in the financial industry, but closely linked to the international business, such as insurance company and shipping company. Eventually, we make contributions on the development in Web services technology and look forward to a bright future in banking and e-business implementations developed by Web services technology. However, as we know, banking has been a high-level security domain all the time. Although we implement WS in this area, some parts of this application are not the actual implementation in a real banking system. What we have done is to simulate those organizations and to focus on developing the WS interfaces that allow them to interact with each other.

To summary, during developing these three Web service applications, we learn and experience the essence of Web services technology. In terms of economics, we can summarize Web services from the following aspects. One hand, Web services are a way to drive down costs by decreasing data and functionality duplication in transaction. Rather than having three departments running three different packages to do the same job because they're all using different systems, the functionality can be centralized and accessed as Web services, regardless of the platform or language each department uses for its own needs. On the other hand, Web services are also a way to drive up income, by allowing an organization to extend their previously purely internal functionality into a wider market and audience. If part of a system does a good job of providing a certain type of valuable information in a timely manner, it could be exposed as a Web service so that it can be accessed as a service by other companies. Therefore, we look forward to a brilliant future on Web services, not only because they are a new renovation in technologies, but also are very powerful tools in e-business or more wider applications to bridge the gap between IT and business.

Appendix

This appendix lists all Visual Basic codes of the project, including ERP module and Web services implementation. Besides, it also lists all SOAP message and WSDL files of each Web service.

A. VB code of ERP Module

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Diagnostics.Process

Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
```

```
Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu
Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem2 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem4 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem5 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem7 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem8 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem9 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem10 As System.Windows.Forms.MenuItem
Friend WithEvents ShippmentGrid As System.Windows.Forms.DataGrid
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents SqlSelectCommand1 As System.Data.SqlClient.SqlCommand
Friend WithEvents SqlInsertCommand1 As System.Data.SqlClient.SqlCommand
Friend WithEvents SqlUpdateCommand1 As System.Data.SqlClient.SqlCommand
Friend WithEvents SqlDeleteCommand1 As System.Data.SqlClient.SqlCommand
Friend WithEvents SqlConnection1 As System.Data.SqlClient.SqlConnection
Friend WithEvents SqlDataAdapter1 As System.Data.SqlClient.SqlDataAdapter
Friend WithEvents QuoteGrid As System.Windows.Forms.DataGrid
Friend WithEvents MenuItem11 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem12 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem13 As System.Windows.Forms.MenuItem
Friend WithEvents EditBox As System.Windows.Forms.Button
Friend WithEvents DeleteBox As System.Windows.Forms.Button
Friend WithEvents AddBox As System.Windows.Forms.Button
Friend WithEvents LCPanel As System.Windows.Forms.Panel
Friend WithEvents RefLabel As System.Windows.Forms.Label
Friend WithEvents RefBox As System.Windows.Forms.TextBox
Friend WithEvents AcctLabel As System.Windows.Forms.Label
Friend WithEvents AmonutLabel As System.Windows.Forms.Label
Friend WithEvents AcctBox As System.Windows.Forms.TextBox
Friend WithEvents AmountBox As System.Windows.Forms.TextBox
Friend WithEvents PreButton As System.Windows.Forms.Button
Friend WithEvents ResultBox As System.Windows.Forms.TextBox
Friend WithEvents ContinueButton As System.Windows.Forms.Button
Friend WithEvents CloseButton As System.Windows.Forms.Button
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents IrateBox As System.Windows.Forms.TextBox
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents InsurancePanel As System.Windows.Forms.Panel
Friend WithEvents MenuItem14 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem15 As System.Windows.Forms.MenuItem
Friend WithEvents BankMenuItem As System.Windows.Forms.MenuItem
```

```
Friend WithEvents InsuranceMenuItem As System.Windows.Forms.MenuItem
Friend WithEvents NameLable As System.Windows.Forms.Label
Friend WithEvents NameBox As System.Windows.Forms.TextBox
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents Label6 As System.Windows.Forms.Label
Friend WithEvents Label7 As System.Windows.Forms.Label
Friend WithEvents Label8 As System.Windows.Forms.Label
Friend WithEvents ToSBox As System.Windows.Forms.TextBox
Friend WithEvents DateBox As System.Windows.Forms.TextBox
Friend WithEvents PoGBox As System.Windows.Forms.TextBox
Friend WithEvents TelphoneBox As System.Windows.Forms.TextBox
Friend WithEvents AddressBox As System.Windows.Forms.TextBox
Friend WithEvents DateCalendar As System.Windows.Forms.MonthCalendar
Friend WithEvents ToSListBox As System.Windows.Forms.ListBox
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label9 As System.Windows.Forms.Label
Friend WithEvents RemarkBox As System.Windows.Forms.TextBox
Friend WithEvents Label10 As System.Windows.Forms.Label
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents Label11 As System.Windows.Forms.Label
Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
Friend WithEvents FeeGrid As System.Windows.Forms.DataGrid
Friend WithEvents FeeLabel As System.Windows.Forms.Label
Friend WithEvents QuoteButton As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.MainMenu1 = New System.Windows.Forms.MainMenu
    Me.MenuItem7 = New System.Windows.Forms.MenuItem
    Me.MenuItem15 = New System.Windows.Forms.MenuItem
    Me.MenuItem1 = New System.Windows.Forms.MenuItem
    Me.MenuItem2 = New System.Windows.Forms.MenuItem
    Me.MenuItem8 = New System.Windows.Forms.MenuItem
    Me.MenuItem9 = New System.Windows.Forms.MenuItem
    Me.MenuItem10 = New System.Windows.Forms.MenuItem
    Me.MenuItem3 = New System.Windows.Forms.MenuItem
    Me.MenuItem11 = New System.Windows.Forms.MenuItem
    Me.MenuItem12 = New System.Windows.Forms.MenuItem
    Me.BankMenuItem = New System.Windows.Forms.MenuItem
    Me.InsuranceMenuItem = New System.Windows.Forms.MenuItem
    Me.MenuItem13 = New System.Windows.Forms.MenuItem
    Me.MenuItem4 = New System.Windows.Forms.MenuItem
    Me.MenuItem5 = New System.Windows.Forms.MenuItem
    Me.MenuItem6 = New System.Windows.Forms.MenuItem
    Me.MenuItem14 = New System.Windows.Forms.MenuItem
```

```
Me.ShippmentGrid = New System.Windows.Forms.DataGrid
Me.Button1 = New System.Windows.Forms.Button
Me.SqlSelectCommand1 = New System.Data.SqlClient.SqlCommand
Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
Me.SqlInsertCommand1 = New System.Data.SqlClient.SqlCommand
Me.SqlUpdateCommand1 = New System.Data.SqlClient.SqlCommand
Me.SqlDeleteCommand1 = New System.Data.SqlClient.SqlCommand
Me.SqlDataAdapter1 = New System.Data.SqlClient.SqlDataAdapter
Me.QuoteGrid = New System.Windows.Forms.DataGrid
Me.EditBox = New System.Windows.Forms.Button
Me.DeleteBox = New System.Windows.Forms.Button
Me.AddBox = New System.Windows.Forms.Button
Me.LCPanel = New System.Windows.Forms.Panel
Me.Label2 = New System.Windows.Forms.Label
Me.IrateBox = New System.Windows.Forms.TextBox
Me.Label1 = New System.Windows.Forms.Label
Me.CloseButton = New System.Windows.Forms.Button
Me.ContinueButton = New System.Windows.Forms.Button
Me.ResultBox = New System.Windows.Forms.TextBox
Me.PreButton = New System.Windows.Forms.Button
Me.AmountBox = New System.Windows.Forms.TextBox
Me.AcctBox = New System.Windows.Forms.TextBox
Me.AmonutLabel = New System.Windows.Forms.Label
Me.AcctLabel = New System.Windows.Forms.Label
Me.RefBox = New System.Windows.Forms.TextBox
Me.RefLabel = New System.Windows.Forms.Label
Me.InsurancePanel = New System.Windows.Forms.Panel
Me.QuoteButton = New System.Windows.Forms.Button
Me.FeeLabel = New System.Windows.Forms.Label
Me.FeeGrid = New System.Windows.Forms.DataGrid
Me.TextBox2 = New System.Windows.Forms.TextBox
Me.Label11 = New System.Windows.Forms.Label
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.Label10 = New System.Windows.Forms.Label
Me.RemarkBox = New System.Windows.Forms.TextBox
Me.Label9 = New System.Windows.Forms.Label
Me.Label3 = New System.Windows.Forms.Label
Me.ToSListBox = New System.Windows.Forms.ListBox
Me.DateCalendar = New System.Windows.Forms.MonthCalendar
Me.AddressBox = New System.Windows.Forms.TextBox
Me.TelphoneBox = New System.Windows.Forms.TextBox
Me.PoGBox = New System.Windows.Forms.TextBox
Me.DateBox = New System.Windows.Forms.TextBox
Me.ToSBox = New System.Windows.Forms.TextBox
```

```
        Me.Label8 = New System.Windows.Forms.Label
        Me.Label7 = New System.Windows.Forms.Label
        Me.Label6 = New System.Windows.Forms.Label
        Me.Label5 = New System.Windows.Forms.Label
        Me.Label4 = New System.Windows.Forms.Label
        Me.NameBox = New System.Windows.Forms.TextBox
        Me.NameLable = New System.Windows.Forms.Label
        CType(Me.ShippmentGrid, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.QuoteGrid, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.LCPanel.SuspendLayout()
        Me.InsurancePanel.SuspendLayout()
        CType(Me.FeeGrid, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'MainMenu1
        '
        Me.MainMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem7,
Me.MenuItem1, Me.MenuItem2, Me.MenuItem3, Me.MenuItem4, Me.MenuItem5, Me.MenuItem6,
Me.MenuItem14})
        '
        'MenuItem7
        '
        Me.MenuItem7.Index = 0
        Me.MenuItem7.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem15})
        Me.MenuItem7.Text = "Administration"
        '
        'MenuItem15
        '
        Me.MenuItem15.Index = 0
        Me.MenuItem15.Text = "Configuration"
        '
        'MenuItem1
        '
        Me.MenuItem1.Index = 1
        Me.MenuItem1.Text = "Production Line"
        '
        'MenuItem2
        '
        Me.MenuItem2.Index = 2
        Me.MenuItem2.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem8,
Me.MenuItem9, Me.MenuItem10})
        Me.MenuItem2.Text = "Inventory"
        '
        'MenuItem8
```

```
        '
        Me.MenuItem8.Index = 0
        Me.MenuItem8.Text = "Order"
        '
        'MenuItem9
        '
        Me.MenuItem9.Index = 1
        Me.MenuItem9.Text = "Shipping"
        '
        'MenuItem10
        '
        Me.MenuItem10.Index = 2
        Me.MenuItem10.Text = "Production Database"
        '
        'MenuItem3
        '
        Me.MenuItem3.Index = 3
        Me.MenuItem3.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.MenuItem11,
Me.MenuItem12, Me.MenuItem13})
        Me.MenuItem3.Text = "Accounting"
        '
        'MenuItem11
        '
        Me.MenuItem11.Index = 0
        Me.MenuItem11.Text = "Check Balance"
        '
        'MenuItem12
        '
        Me.MenuItem12.Index = 1
        Me.MenuItem12.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.BankMenuItem,
Me.InsuranceMenuItem})
        Me.MenuItem12.Text = "Financial Support"
        '
        'BankMenuItem
        '
        Me.BankMenuItem.Index = 0
        Me.BankMenuItem.Text = "Bank"
        '
        'InsuranceMenuItem
        '
        Me.InsuranceMenuItem.Index = 1
        Me.InsuranceMenuItem.Text = "Insurance"
        '
        'MenuItem13
```

```
'
Me.MenuItem13.Index = 2
Me.MenuItem13.Text = "Payroll"
'
'MenuItem4
'
Me.MenuItem4.Index = 4
Me.MenuItem4.Text = "Management"
'
'MenuItem5
'
Me.MenuItem5.Index = 5
Me.MenuItem5.Text = "Human Resource"
'
'MenuItem6
'
Me.MenuItem6.Index = 6
Me.MenuItem6.Text = "Reporting"
'
'MenuItem14
'
Me.MenuItem14.Index = 7
Me.MenuItem14.Text = "Help"
'
'ShippmentGrid
'
Me.ShippmentGrid.AllowDrop = True
Me.ShippmentGrid.CaptionVisible = False
Me.ShippmentGrid.DataMember = ""
Me.ShippmentGrid.GridLineStyle = System.Windows.Forms.DataGridLineStyle.None
Me.ShippmentGrid.HeaderForeColor = System.Drawing.SystemColors.ControlText
Me.ShippmentGrid.Location = New System.Drawing.Point(56, 24)
Me.ShippmentGrid.Name = "ShippmentGrid"
Me.ShippmentGrid.ParentRowsVisible = False
Me.ShippmentGrid.ReadOnly = True
Me.ShippmentGrid.Size = New System.Drawing.Size(616, 144)
Me.ShippmentGrid.TabIndex = 0
Me.ShippmentGrid.TabStop = False
Me.ShippmentGrid.Visible = False
'
'Button1
'
Me.Button1.Location = New System.Drawing.Point(440, 192)
Me.Button1.Name = "Button1"
```

```
        Me.Button1.Size = New System.Drawing.Size(216, 23)
        Me.Button1.TabIndex = 1
        Me.Button1.Text = "Get Shipping Quote Now!"
        Me.Button1.Visible = False
        '
        'SqlSelectCommand1
        '
        Me.SqlSelectCommand1.CommandText = "SELECT Name, SerialNo, Quantity, Price, ShippingDate,
TofShipp, Description, Loca" & _
        "tion FROM tblOrder"
        Me.SqlSelectCommand1.Connection = Me.SqlConnection1
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=HOME1;packet size=4096;integrated
security=SSPI;data source=HOME1;" & _
        "persist security info=True;initial catalog=ERP"
        '
        'SqlInsertCommand1
        '
        Me.SqlInsertCommand1.CommandText = "INSERT INTO tblOrder(Name, SerialNo, Quantity, Price,
ShippingDate, TofShipp, Des" & _
        "cription, Location) VALUES (@Name, @SerialNo, @Quantity, @Price, @ShippingDate, " & _
        "@TofShipp, @Description, @Location); SELECT Name, SerialNo, Quantity, Price, Shi" & _
        "ppingDate, TofShipp, Description, Location FROM tblOrder WHERE (SerialNo = @Seri" & _
        "alNo)"
        Me.SqlInsertCommand1.Connection = Me.SqlConnection1
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Name",
System.Data.SqlDbType.VarChar, 10, "Name"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@SerialNo",
System.Data.SqlDbType.VarChar, 10, "SerialNo"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Quantity",
System.Data.SqlDbType.VarChar, 10, "Quantity"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Price",
System.Data.SqlDbType.VarChar, 10, "Price"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@ShippingDate",
System.Data.SqlDbType.DateTime, 8, "ShippingDate"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@TofShipp",
System.Data.SqlDbType.VarChar, 10, "TofShipp"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Description",
System.Data.SqlDbType.VarChar, 10, "Description"))
        Me.SqlInsertCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Location",
System.Data.SqlDbType.VarChar, 10, "Location"))
        '
```

```
        'SqlUpdateCommand1
        '
        Me.SqlUpdateCommand1.CommandText = "UPDATE tblOrder SET Name = @Name, SerialNo =
@SerialNo, Quantity = @Quantity, Pri" & _
        "ce = @Price, ShippingDate = @ShippingDate, TofShipp = @TofShipp, Description = @" & _
        "Description, Location = @Location WHERE (SerialNo = @Original_SerialNo) AND (Des" & _
        "cription = @Original_Description OR @Original_Description IS NULL AND Descriptio" & _
        "n IS NULL) AND (Location = @Original_Location OR @Original_Location IS NULL AND " & _
        "Location IS NULL) AND (Name = @Original_Name OR @Original_Name IS NULL AND Name " &
_
        "IS NULL) AND (Price = @Original_Price OR @Original_Price IS NULL AND Price IS NU" & _
        "LL) AND (Quantity = @Original_Quantity OR @Original_Quantity IS NULL AND Quantit" & _
        "y IS NULL) AND (ShippingDate = @Original_ShippingDate OR @Original_ShippingDate " & _
        "IS NULL AND ShippingDate IS NULL) AND (TofShipp = @Original_TofShipp OR @Origina" & _
        "l_TofShipp IS NULL AND TofShipp IS NULL); SELECT Name, SerialNo, Quantity, Price" & _
        ", ShippingDate, TofShipp, Description, Location FROM tblOrder WHERE (SerialNo = " & _
        "@SerialNo)"
        Me.SqlUpdateCommand1.Connection = Me.SqlConnection1
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Name",
System.Data.SqlDbType.VarChar, 10, "Name"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@SerialNo",
System.Data.SqlDbType.VarChar, 10, "SerialNo"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Quantity",
System.Data.SqlDbType.VarChar, 10, "Quantity"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Price",
System.Data.SqlDbType.VarChar, 10, "Price"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@ShippingDate",
System.Data.SqlDbType.DateTime, 8, "ShippingDate"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@TofShipp",
System.Data.SqlDbType.VarChar, 10, "TofShipp"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Description",
System.Data.SqlDbType.VarChar, 10, "Description"))
        Me.SqlUpdateCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Location",
System.Data.SqlDbType.VarChar, 10, "Location"))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_SerialNo", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "SerialNo",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Description", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Description",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Location", System.Data.SqlDbType.VarChar, 10,
```

```
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Location",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Name", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Name",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Price", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Price",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Quantity", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Quantity",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_ShippingDate", System.Data.SqlDbType.DateTime, 8,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "ShippingDate",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlUpdateCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_TofShipp", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "TofShipp",
System.Data.DataRowVersion.Original, Nothing))
        '
        'SqlDeleteCommand1
        '
        Me.SqlDeleteCommand1.CommandText = "DELETE FROM tblOrder WHERE (SerialNo =
@Original_SerialNo) AND (Description = @Or" & _
        "iginal_Description OR @Original_Description IS NULL AND Description IS NULL) AND" & _
        " (Location = @Original_Location OR @Original_Location IS NULL AND Location IS NU" & _
        "LL) AND (Name = @Original_Name OR @Original_Name IS NULL AND Name IS NULL) AND ("
& _
        "Price = @Original_Price OR @Original_Price IS NULL AND Price IS NULL) AND (Quant" & _
        "ity = @Original_Quantity OR @Original_Quantity IS NULL AND Quantity IS NULL) AND" & _
        " (ShippingDate = @Original_ShippingDate OR @Original_ShippingDate IS NULL AND Sh" & _
        "ippingDate IS NULL) AND (TofShipp = @Original_TofShipp OR @Original_TofShipp IS " & _
        "NULL AND TofShipp IS NULL)"
        Me.SqlDeleteCommand1.Connection = Me.SqlConnection1
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_SerialNo", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "SerialNo",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Description", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Description",
```

```
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Location", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Location",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Name", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Name",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New System.Data.SqlClient.SqlParameter("@Original_Price",
System.Data.SqlDbType.VarChar, 10, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0,
Byte), "Price", System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_Quantity", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Quantity",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_ShippingDate", System.Data.SqlDbType.DateTime, 8,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "ShippingDate",
System.Data.DataRowVersion.Original, Nothing))
        Me.SqlDeleteCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@Original_TofShipp", System.Data.SqlDbType.VarChar, 10,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "TofShipp",
System.Data.DataRowVersion.Original, Nothing))
        '
        'SqlDataAdapter1
        '
        Me.SqlDataAdapter1.DeleteCommand = Me.SqlDeleteCommand1
        Me.SqlDataAdapter1.InsertCommand = Me.SqlInsertCommand1
        Me.SqlDataAdapter1.SelectCommand = Me.SqlSelectCommand1
        Me.SqlDataAdapter1.TableMappings.AddRange(New System.Data.Common.DataTableMapping()
{New System.Data.Common.DataTableMapping("Table", "tblOrder", New
System.Data.Common.DataColumnMapping() {New System.Data.Common.DataColumnMapping("Name",
"Name"), New System.Data.Common.DataColumnMapping("SerialNo", "SerialNo"), New
System.Data.Common.DataColumnMapping("Quantity", "Quantity"), New
System.Data.Common.DataColumnMapping("Price", "Price"), New
System.Data.Common.DataColumnMapping("ShippingDate", "ShippingDate"), New
System.Data.Common.DataColumnMapping("TofShipp", "TofShipp"), New
System.Data.Common.DataColumnMapping("Description", "Description"), New
System.Data.Common.DataColumnMapping("Location", "Location")})})
        Me.SqlDataAdapter1.UpdateCommand = Me.SqlUpdateCommand1
        '
        'QuoteGrid
        '
```

```
Me.QuoteGrid.CaptionVisible = False
Me.QuoteGrid.DataMember = ""
Me.QuoteGrid.HeaderForeColor = System.Drawing.SystemColors.ControlText
Me.QuoteGrid.Location = New System.Drawing.Point(168, 256)
Me.QuoteGrid.Name = "QuoteGrid"
Me.QuoteGrid.ParentRowsVisible = False
Me.QuoteGrid.PreferredColumnWidth = 100
Me.QuoteGrid.Size = New System.Drawing.Size(360, 96)
Me.QuoteGrid.TabIndex = 2
Me.QuoteGrid.Visible = False
'
'EditBox
'
Me.EditBox.Location = New System.Drawing.Point(64, 192)
Me.EditBox.Name = "EditBox"
Me.EditBox.TabIndex = 3
Me.EditBox.Text = "Edit"
Me.EditBox.Visible = False
'
'DeleteBox
'
Me.DeleteBox.Location = New System.Drawing.Point(320, 192)
Me.DeleteBox.Name = "DeleteBox"
Me.DeleteBox.TabIndex = 5
Me.DeleteBox.Text = "Delete"
Me.DeleteBox.Visible = False
'
'AddBox
'
Me.AddBox.Location = New System.Drawing.Point(192, 192)
Me.AddBox.Name = "AddBox"
Me.AddBox.TabIndex = 6
Me.AddBox.Text = "Add"
Me.AddBox.Visible = False
'
'LCPanel
'
Me.LCPanel.Controls.Add(Me.Label2)
Me.LCPanel.Controls.Add(Me.IrateBox)
Me.LCPanel.Controls.Add(Me.Label1)
Me.LCPanel.Controls.Add(Me.CloseButton)
Me.LCPanel.Controls.Add(Me.ContinueButton)
Me.LCPanel.Controls.Add(Me.ResultBox)
Me.LCPanel.Controls.Add(Me.PreButton)
```

```
Me.LCPanel.Controls.Add(Me.AmountBox)
Me.LCPanel.Controls.Add(Me.AcctBox)
Me.LCPanel.Controls.Add(Me.AmonutLabel)
Me.LCPanel.Controls.Add(Me.AcctLabel)
Me.LCPanel.Controls.Add(Me.RefBox)
Me.LCPanel.Controls.Add(Me.RefLabel)
Me.LCPanel.Location = New System.Drawing.Point(120, 72)
Me.LCPanel.Name = "LCPanel"
Me.LCPanel.Size = New System.Drawing.Size(440, 296)
Me.LCPanel.TabIndex = 7
Me.LCPanel.Visible = False
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(328, 120)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(40, 23)
Me.Label2.TabIndex = 16
Me.Label2.Text = "%"
'
'IrateBox
'
Me.IrateBox.Location = New System.Drawing.Point(280, 112)
Me.IrateBox.Name = "IrateBox"
Me.IrateBox.Size = New System.Drawing.Size(40, 21)
Me.IrateBox.TabIndex = 15
Me.IrateBox.Text = ""
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(24, 112)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(248, 23)
Me.Label1.TabIndex = 14
Me.Label1.Text = "Your interest rate expectaion is under:"
'
'CloseButton
'
Me.CloseButton.Location = New System.Drawing.Point(344, 232)
Me.CloseButton.Name = "CloseButton"
Me.CloseButton.TabIndex = 13
Me.CloseButton.Text = "Close"
Me.CloseButton.Visible = False
'
```

```
        'ContinueButton
        '
        Me.ContinueButton.Location = New System.Drawing.Point(344, 184)
        Me.ContinueButton.Name = "ContinueButton"
        Me.ContinueButton.TabIndex = 12
        Me.ContinueButton.Text = "Continue"
        Me.ContinueButton.Visible = False
        '
        'ResultBox
        '
        Me.ResultBox.AllowDrop = True
        Me.ResultBox.Font = New System.Drawing.Font("SimSun", 10.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(134, Byte))
        Me.ResultBox.Location = New System.Drawing.Point(32, 160)
        Me.ResultBox.Multiline = True
        Me.ResultBox.Name = "ResultBox"
        Me.ResultBox.Size = New System.Drawing.Size(288, 112)
        Me.ResultBox.TabIndex = 11
        Me.ResultBox.Text = ""
        Me.ResultBox.Visible = False
        '
        'PreButton
        '
        Me.PreButton.Location = New System.Drawing.Point(328, 64)
        Me.PreButton.Name = "PreButton"
        Me.PreButton.TabIndex = 10
        Me.PreButton.Text = "Prerequest"
        '
        'AmountBox
        '
        Me.AmountBox.Location = New System.Drawing.Point(328, 24)
        Me.AmountBox.Name = "AmountBox"
        Me.AmountBox.Size = New System.Drawing.Size(80, 21)
        Me.AmountBox.TabIndex = 9
        Me.AmountBox.Text = ""
        '
        'AcctBox
        '
        Me.AcctBox.Location = New System.Drawing.Point(152, 64)
        Me.AcctBox.Name = "AcctBox"
        Me.AcctBox.TabIndex = 8
        Me.AcctBox.Text = ""
        '
        'AmonutLabel
```

```
'
Me.AmonutLabel.Location = New System.Drawing.Point(280, 24)
Me.AmonutLabel.Name = "AmonutLabel"
Me.AmonutLabel.Size = New System.Drawing.Size(48, 23)
Me.AmonutLabel.TabIndex = 6
Me.AmonutLabel.Text = "Amount"
'
'AcctLabel
'
Me.AcctLabel.Location = New System.Drawing.Point(24, 64)
Me.AcctLabel.Name = "AcctLabel"
Me.AcctLabel.TabIndex = 3
Me.AcctLabel.Text = "Account Number"
'
'RefBox
'
Me.RefBox.Location = New System.Drawing.Point(152, 24)
Me.RefBox.Name = "RefBox"
Me.RefBox.Size = New System.Drawing.Size(104, 21)
Me.RefBox.TabIndex = 1
Me.RefBox.Text = ""
'
'RefLabel
'
Me.RefLabel.Location = New System.Drawing.Point(24, 24)
Me.RefLabel.Name = "RefLabel"
Me.RefLabel.Size = New System.Drawing.Size(112, 23)
Me.RefLabel.TabIndex = 0
Me.RefLabel.Text = "Reference Number"
'
'InsurancePanel
'
Me.InsurancePanel.Controls.Add(Me.QuoteButton)
Me.InsurancePanel.Controls.Add(Me.FeeLabel)
Me.InsurancePanel.Controls.Add(Me.FeeGrid)
Me.InsurancePanel.Controls.Add(Me.TextBox2)
Me.InsurancePanel.Controls.Add(Me.Label11)
Me.InsurancePanel.Controls.Add(Me.TextBox1)
Me.InsurancePanel.Controls.Add(Me.Label10)
Me.InsurancePanel.Controls.Add(Me.RemarkBox)
Me.InsurancePanel.Controls.Add(Me.Label9)
Me.InsurancePanel.Controls.Add(Me.Label3)
Me.InsurancePanel.Controls.Add(Me.ToSListBox)
Me.InsurancePanel.Controls.Add(Me.DateCalendar)
```

```
        Me.InsurancePanel.Controls.Add(Me.AddressBox)
        Me.InsurancePanel.Controls.Add(Me.TelphoneBox)
        Me.InsurancePanel.Controls.Add(Me.PoGBox)
        Me.InsurancePanel.Controls.Add(Me.DateBox)
        Me.InsurancePanel.Controls.Add(Me.ToSBox)
        Me.InsurancePanel.Controls.Add(Me.Label8)
        Me.InsurancePanel.Controls.Add(Me.Label7)
        Me.InsurancePanel.Controls.Add(Me.Label6)
        Me.InsurancePanel.Controls.Add(Me.Label5)
        Me.InsurancePanel.Controls.Add(Me.Label4)
        Me.InsurancePanel.Controls.Add(Me.NameBox)
        Me.InsurancePanel.Controls.Add(Me.NameLable)
        Me.InsurancePanel.Location = New System.Drawing.Point(56, 48)
        Me.InsurancePanel.Name = "InsurancePanel"
        Me.InsurancePanel.Size = New System.Drawing.Size(616, 296)
        Me.InsurancePanel.TabIndex = 17
        Me.InsurancePanel.Visible = False
        '
        'QuoteButton
        '
        Me.QuoteButton.Location = New System.Drawing.Point(192, 248)
        Me.QuoteButton.Name = "QuoteButton"
        Me.QuoteButton.Size = New System.Drawing.Size(80, 24)
        Me.QuoteButton.TabIndex = 24
        Me.QuoteButton.Text = "Quote Now"
        '
        'FeeLabel
        '
        Me.FeeLabel.Font = New System.Drawing.Font("SimSun", 12.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(134, Byte))
        Me.FeeLabel.Location = New System.Drawing.Point(392, 136)
        Me.FeeLabel.Name = "FeeLabel"
        Me.FeeLabel.Size = New System.Drawing.Size(176, 24)
        Me.FeeLabel.TabIndex = 23
        Me.FeeLabel.Text = "Your Insurance Fee:"
        Me.FeeLabel.Visible = False
        '
        'FeeGrid
        '
        Me.FeeGrid.CaptionVisible = False
        Me.FeeGrid.DataMember = ""
        Me.FeeGrid.HeaderForeColor = System.Drawing.SystemColors.ControlText
        Me.FeeGrid.Location = New System.Drawing.Point(392, 168)
        Me.FeeGrid.Name = "FeeGrid"
```

```
Me.FeeGrid.ParentRowsVisible = False
Me.FeeGrid.Size = New System.Drawing.Size(208, 88)
Me.FeeGrid.TabIndex = 22
Me.FeeGrid.Visible = False
'
'TextBox2
'
Me.TextBox2.Location = New System.Drawing.Point(304, 96)
Me.TextBox2.Name = "TextBox2"
Me.TextBox2.TabIndex = 21
Me.TextBox2.Text = ""
'
'Label11
'
Me.Label11.Location = New System.Drawing.Point(232, 104)
Me.Label11.Name = "Label11"
Me.Label11.Size = New System.Drawing.Size(48, 23)
Me.Label11.TabIndex = 20
Me.Label11.Text = "Ship To"
'
'TextBox1
'
Me.TextBox1.Location = New System.Drawing.Point(104, 96)
Me.TextBox1.Name = "TextBox1"
Me.TextBox1.TabIndex = 19
Me.TextBox1.Text = ""
'
'Label10
'
Me.Label10.Location = New System.Drawing.Point(8, 104)
Me.Label10.Name = "Label10"
Me.Label10.Size = New System.Drawing.Size(64, 23)
Me.Label10.TabIndex = 18
Me.Label10.Text = "Ship From"
'
'RemarkBox
'
Me.RemarkBox.Location = New System.Drawing.Point(96, 136)
Me.RemarkBox.Multiline = True
Me.RemarkBox.Name = "RemarkBox"
Me.RemarkBox.Size = New System.Drawing.Size(272, 96)
Me.RemarkBox.TabIndex = 17
Me.RemarkBox.Text = ""
'
```

```
'Label9
'
Me.Label9.Location = New System.Drawing.Point(24, 136)
Me.Label9.Name = "Label9"
Me.Label9.Size = New System.Drawing.Size(64, 23)
Me.Label9.TabIndex = 16
Me.Label9.Text = "Remark"
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(424, 96)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(88, 23)
Me.Label3.TabIndex = 15
Me.Label3.Text = "Shipping Date"
'
'ToSListBox
'
Me.ToSListBox.ItemHeight = 12
Me.ToSListBox.Items.AddRange(New Object() {"Air", "Land", "Sea"})
Me.ToSListBox.Location = New System.Drawing.Point(528, 48)
Me.ToSListBox.Name = "ToSListBox"
Me.ToSListBox.Size = New System.Drawing.Size(72, 28)
Me.ToSListBox.TabIndex = 14
'
'DateCalendar
'
Me.DateCalendar.Location = New System.Drawing.Point(392, 136)
Me.DateCalendar.Name = "DateCalendar"
Me.DateCalendar.TabIndex = 13
'
'AddressBox
'
Me.AddressBox.Location = New System.Drawing.Point(264, 8)
Me.AddressBox.Name = "AddressBox"
Me.AddressBox.ReadOnly = True
Me.AddressBox.Size = New System.Drawing.Size(168, 21)
Me.AddressBox.TabIndex = 11
Me.AddressBox.Text = "728 Sheppard St. Toronto"
'
'TelphoneBox
'
Me.TelphoneBox.Location = New System.Drawing.Point(520, 8)
Me.TelphoneBox.Name = "TelphoneBox"
```

```
Me.TelphoneBox.ReadOnly = True
Me.TelphoneBox.Size = New System.Drawing.Size(80, 21)
Me.TelphoneBox.TabIndex = 10
Me.TelphoneBox.Text = "416-866-6666"
'
'PoGBox
'
Me.PoGBox.Location = New System.Drawing.Point(344, 48)
Me.PoGBox.Name = "PoGBox"
Me.PoGBox.Size = New System.Drawing.Size(48, 21)
Me.PoGBox.TabIndex = 9
Me.PoGBox.Text = ""
'
'DateBox
'
Me.DateBox.Location = New System.Drawing.Point(528, 96)
Me.DateBox.Name = "DateBox"
Me.DateBox.Size = New System.Drawing.Size(72, 21)
Me.DateBox.TabIndex = 8
Me.DateBox.Text = ""
'
'ToSBox
'
Me.ToSBox.Location = New System.Drawing.Point(112, 48)
Me.ToSBox.Name = "ToSBox"
Me.ToSBox.Size = New System.Drawing.Size(72, 21)
Me.ToSBox.TabIndex = 7
Me.ToSBox.Text = ""
'
'Label8
'
Me.Label8.Location = New System.Drawing.Point(408, 56)
Me.Label8.Name = "Label8"
Me.Label8.Size = New System.Drawing.Size(104, 23)
Me.Label8.TabIndex = 6
Me.Label8.Text = "Type of Shipment"
'
'Label7
'
Me.Label7.Location = New System.Drawing.Point(8, 56)
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(88, 23)
Me.Label7.TabIndex = 5
Me.Label7.Text = "Type of Goods"
```

```
'
'Label6
'
Me.Label6.Location = New System.Drawing.Point(200, 56)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(128, 23)
Me.Label6.TabIndex = 4
Me.Label6.Text = "Price of Goods (CAD)"
'
'Label5
'
Me.Label5.Location = New System.Drawing.Point(456, 16)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(56, 23)
Me.Label5.TabIndex = 3
Me.Label5.Text = "Telphone"
'
'Label4
'
Me.Label4.Location = New System.Drawing.Point(208, 16)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(56, 23)
Me.Label4.TabIndex = 2
Me.Label4.Text = "Address"
'
'NameBox
'
Me.NameBox.Location = New System.Drawing.Point(40, 8)
Me.NameBox.Name = "NameBox"
Me.NameBox.ReadOnly = True
Me.NameBox.Size = New System.Drawing.Size(144, 21)
Me.NameBox.TabIndex = 1
Me.NameBox.Text = "Infinity Energy Ltd."
'
'NameLable
'
Me.NameLable.Location = New System.Drawing.Point(8, 16)
Me.NameLable.Name = "NameLable"
Me.NameLable.Size = New System.Drawing.Size(32, 23)
Me.NameLable.TabIndex = 0
Me.NameLable.Text = "Name"
'
'Form1
'
```

```
        Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
        Me.ClientSize = New System.Drawing.Size(728, 401)
        Me.Controls.Add(Me.LCPanel)
        Me.Controls.Add(Me.AddBox)
        Me.Controls.Add(Me.DeleteBox)
        Me.Controls.Add(Me.EditBox)
        Me.Controls.Add(Me.QuoteGrid)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.ShippmentGrid)
        Me.Controls.Add(Me.InsurancePanel)
        Me.Menu = Me.MainMenu1
        Me.Name = "Form1"
        Me.Text = "Main Manu"
        CType(Me.ShippmentGrid, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.QuoteGrid, System.ComponentModel.ISupportInitialize).EndInit()
        Me.LCPanel.ResumeLayout(False)
        Me.InsurancePanel.ResumeLayout(False)
        CType(Me.FeeGrid, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub

#End Region

    Dim row As Integer
    Private Agent As New localhost.Service1

    Private Sub MenuItem9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MenuItem9.Click
        Dim dataadapter As SqlDataAdapter
        Dim connection As SqlConnection
        Dim str As String
        Dim ds As DataSet
        Dim reader As SqlDataReader

        LCPanel.Visible = False
        InsurancePanel.Visible = False
        ShippmentGrid.Visible = True
        Button1.Visible = True
        QuoteGrid.Visible = False
        AddBox.Visible = True
        EditBox.Visible = True
        DeleteBox.Visible = True
```

```
        str = "select * from tblOrder"
        connection = New SqlConnection("server= localhost; uid=sa;pwd=; database=ERP")
        ' connect to database
        dataadapter = New SqlDataAdapter(str, connection)
        connection.Open()
        ' reader = dataadapter.SelectCommand.ExecuteReader()
        ds = New DataSet
        dataadapter.Fill(ds)
        connection.Close()
        ShippmentGrid.DataSource = ds
    End Sub

    Private Sub ShippmentGrid_Navigate(ByVal sender As System.Object, ByVal ne As
System.Windows.Forms.NavigateEventArgs) Handles ShippmentGrid.Navigate
        row = ShippmentGrid.CurrentRowIndex
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim quantity As Integer
        Dim size As Short
        Dim type As String
        Dim str As String
        Dim conn As SqlConnection
        Dim myadapter As SqlDataAdapter
        Dim ds As DataSet
        Dim result As Boolean

        quantity = ShippmentGrid.Item(row, 2)
        size = ShippmentGrid.Item(row, 7) * quantity
        type = ShippmentGrid.Item(row, 5)
        type = type.Trim
        result = Agent.ShippInterface(quantity.ToString, type, size)
        If result Then
            QuoteGrid.Visible = True
            str = "select * from tblQuotes"
            conn = New SqlConnection("server= localhost; uid=sa;pwd=; database=ERP")
            ' connect to database
            myadapter = New SqlDataAdapter(str, conn)
            conn.Open()
            ' reader = dataadapter.SelectCommand.ExecuteReader()
            ds = New DataSet
            myadapter.Fill(ds)
            conn.Close()
```

```
            QuoteGrid.DataSource = ds
        End If
    End Sub

    Private Agent1 As New localhost1.Apply

    Private Sub PreButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
PreButton.Click
        Dim result As Boolean
        Dim amount As Long
        Dim rate As Decimal

        ResultBox.Visible = True
        amount = Val(AmountBox.Text)
        result = Agent1.ApplyLC(RefBox.Text, amount, AcctBox.Text).boo
        ResultBox.Text = Agent1.ApplyLC(RefBox.Text, amount, AcctBox.Text).str
        rate = Agent1.ApplyLC(RefBox.Text, amount, AcctBox.Text).i
        If result Then
            ContinueButton.Visible = True
        Else
            If rate <= CShort(IrateBox.Text) / 100 Then
                ContinueButton.Visible = True
            End If
            CloseButton.Visible = True
        End If
    End Sub

    Private Sub ContinueButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ContinueButton.Click
        Dim Form2 As New Form2

        Form2.RefBox.Text = RefBox.Text
        Form2.AmountBox.Text = AmountBox.Text
        Form2.AcctBox.Text = AcctBox.Text
        Form2.ApplyButton.Visible = True
        Form2.Show()
        ContinueButton.Visible = False
        ResultBox.Visible = False
    End Sub

    Private Sub CloseButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CloseButton.Click
        ResultBox.Visible = False
        ContinueButton.Visible = False
```

```
        CloseButton.Visible = False
    End Sub

    Private Sub BankMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BankMenuItem.Click
        ShippmentGrid.Visible = False
        Button1.Visible = False
        QuoteGrid.Visible = False
        AddBox.Visible = False
        EditBox.Visible = False
        DeleteBox.Visible = False
        QuoteGrid.Visible = False
        InsurancePanel.Visible = False
        LCPanel.Visible = True
    End Sub

    Private Agent3 As New localhostInsurance.InsuranceInterface

    Private Sub QuoteButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
QuoteButton.Click
        Dim conn As SqlConnection
        Dim myadapter As SqlDataAdapter
        Dim ds As DataSet
        Dim price As String
        Dim type As String
        Dim str As String

        price = PoGBox.Text
        type = ToSListBox.SelectedItem
        If Agent3.InsuranceInterface(price, type) Then
            FeeLabel.Visible = True
            DateCalendar.Visible = False
            FeeGrid.Visible = True
            str = "select * from tblInsurance"
            conn = New SqlConnection("server= localhost; uid=sa;pwd=; database=ERP")
            ' connect to database
            myadapter = New SqlDataAdapter(str, conn)
            conn.Open()
            ' reader = dataadapter.SelectCommand.ExecuteReader()
            ds = New DataSet
            myadapter.Fill(ds)
            conn.Close()
            FeeGrid.DataSource = ds
        End If
```

```
    End Sub

    Private Sub DateCalendar_DateChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DateRangeEventArgs) Handles DateCalendar.DateChanged
        DateBox.Text = DateCalendar.SelectionStart
    End Sub

    Private Sub InsuranceMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles InsuranceMenuItem.Click
        InsurancePanel.Visible = True
        ShippmentGrid.Visible = False
        Button1.Visible = False
        QuoteGrid.Visible = False
        AddBox.Visible = False
        EditBox.Visible = False
        DeleteBox.Visible = False
        QuoteGrid.Visible = False
        LCPanel.Visible = False
    End Sub
End Class

Public Class Form2
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
```

```
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents Label6 As System.Windows.Forms.Label
Friend WithEvents Label7 As System.Windows.Forms.Label
Friend WithEvents Label8 As System.Windows.Forms.Label
Friend WithEvents Label9 As System.Windows.Forms.Label
Friend WithEvents ExpireDate As System.Windows.Forms.TextBox
Friend WithEvents IssueBox As System.Windows.Forms.TextBox
Friend WithEvents NotBox As System.Windows.Forms.TextBox
Friend WithEvents NegBox As System.Windows.Forms.TextBox
Friend WithEvents PayBox As System.Windows.Forms.TextBox
Friend WithEvents ContractBox As System.Windows.Forms.TextBox
Friend WithEvents AmountBox As System.Windows.Forms.TextBox
Friend WithEvents AcctBox As System.Windows.Forms.TextBox
Friend WithEvents RefBox As System.Windows.Forms.TextBox
Friend WithEvents ApplyButton As System.Windows.Forms.Button
Friend WithEvents ExpireCalendar As System.Windows.Forms.MonthCalendar
Friend WithEvents Label11 As System.Windows.Forms.Label
Friend WithEvents MemoBox As System.Windows.Forms.TextBox
Friend WithEvents ResultPanel As System.Windows.Forms.Panel
Friend WithEvents label As System.Windows.Forms.Label
Friend WithEvents LCNoBox As System.Windows.Forms.TextBox
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.GroupBox1 = New System.Windows.Forms.GroupBox
    Me.ResultPanel = New System.Windows.Forms.Panel
    Me.LCNoBox = New System.Windows.Forms.TextBox
    Me.label = New System.Windows.Forms.Label
    Me.MemoBox = New System.Windows.Forms.TextBox
    Me.Label11 = New System.Windows.Forms.Label
    Me.ExpireCalendar = New System.Windows.Forms.MonthCalendar
    Me.Label9 = New System.Windows.Forms.Label
    Me.Label8 = New System.Windows.Forms.Label
```

```
Me.Label7 = New System.Windows.Forms.Label
Me.Label6 = New System.Windows.Forms.Label
Me.Label5 = New System.Windows.Forms.Label
Me.Label4 = New System.Windows.Forms.Label
Me.Label3 = New System.Windows.Forms.Label
Me.Label2 = New System.Windows.Forms.Label
Me.Label1 = New System.Windows.Forms.Label
Me.ExpireDate = New System.Windows.Forms.TextBox
Me.IssueBox = New System.Windows.Forms.TextBox
Me.NotBox = New System.Windows.Forms.TextBox
Me.NegBox = New System.Windows.Forms.TextBox
Me.PayBox = New System.Windows.Forms.TextBox
Me.ContractBox = New System.Windows.Forms.TextBox
Me.AmountBox = New System.Windows.Forms.TextBox
Me.AcctBox = New System.Windows.Forms.TextBox
Me.RefBox = New System.Windows.Forms.TextBox
Me.ApplyButton = New System.Windows.Forms.Button
Me.GroupBox1.SuspendLayout()
Me.ResultPanel.SuspendLayout()
Me.SuspendLayout()
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.ResultPanel)
Me.GroupBox1.Controls.Add(Me.MemoBox)
Me.GroupBox1.Controls.Add(Me.Label11)
Me.GroupBox1.Controls.Add(Me.ExpireCalendar)
Me.GroupBox1.Controls.Add(Me.Label9)
Me.GroupBox1.Controls.Add(Me.Label8)
Me.GroupBox1.Controls.Add(Me.Label7)
Me.GroupBox1.Controls.Add(Me.Label6)
Me.GroupBox1.Controls.Add(Me.Label5)
Me.GroupBox1.Controls.Add(Me.Label4)
Me.GroupBox1.Controls.Add(Me.Label3)
Me.GroupBox1.Controls.Add(Me.Label2)
Me.GroupBox1.Controls.Add(Me.Label1)
Me.GroupBox1.Controls.Add(Me.ExpireDate)
Me.GroupBox1.Controls.Add(Me.IssueBox)
Me.GroupBox1.Controls.Add(Me.NotBox)
Me.GroupBox1.Controls.Add(Me.NegBox)
Me.GroupBox1.Controls.Add(Me.PayBox)
Me.GroupBox1.Controls.Add(Me.ContractBox)
Me.GroupBox1.Controls.Add(Me.AmountBox)
Me.GroupBox1.Controls.Add(Me.AcctBox)
```

```
Me.GroupBox1.Controls.Add(Me.RefBox)
Me.GroupBox1.Controls.Add(Me.ApplyButton)
Me.GroupBox1.Location = New System.Drawing.Point(24, 16)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(568, 368)
Me.GroupBox1.TabIndex = 0
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Please enter your information:"
'
'ResultPanel
'
Me.ResultPanel.Controls.Add(Me.LCNoBox)
Me.ResultPanel.Controls.Add(Me.label)
Me.ResultPanel.Location = New System.Drawing.Point(272, 208)
Me.ResultPanel.Name = "ResultPanel"
Me.ResultPanel.Size = New System.Drawing.Size(264, 72)
Me.ResultPanel.TabIndex = 24
Me.ResultPanel.Visible = False
'
'LCNoBox
'
Me.LCNoBox.Location = New System.Drawing.Point(88, 24)
Me.LCNoBox.Name = "LCNoBox"
Me.LCNoBox.Size = New System.Drawing.Size(168, 21)
Me.LCNoBox.TabIndex = 1
Me.LCNoBox.Text = ""
'
'label
'
Me.label.Location = New System.Drawing.Point(8, 24)
Me.label.Name = "label"
Me.label.Size = New System.Drawing.Size(80, 23)
Me.label.TabIndex = 0
Me.label.Text = "L/C Number"
'
'MemoBox
'
Me.MemoBox.Location = New System.Drawing.Point(88, 176)
Me.MemoBox.Multiline = True
Me.MemoBox.Name = "MemoBox"
Me.MemoBox.Size = New System.Drawing.Size(160, 128)
Me.MemoBox.TabIndex = 23
Me.MemoBox.Text = ""
'
```

```
'Label11
'
Me.Label11.Location = New System.Drawing.Point(32, 176)
Me.Label11.Name = "Label11"
Me.Label11.Size = New System.Drawing.Size(56, 23)
Me.Label11.TabIndex = 22
Me.Label11.Text = "Memo"
'
'ExpireCalendar
'
Me.ExpireCalendar.AllowDrop = True
Me.ExpireCalendar.Location = New System.Drawing.Point(328, 208)
Me.ExpireCalendar.Name = "ExpireCalendar"
Me.ExpireCalendar.ShowTodayCircle = False
Me.ExpireCalendar.TabIndex = 21
'
'Label9
'
Me.Label9.Location = New System.Drawing.Point(272, 176)
Me.Label9.Name = "Label9"
Me.Label9.Size = New System.Drawing.Size(72, 23)
Me.Label9.TabIndex = 19
Me.Label9.Text = "Expire Date"
'
'Label8
'
Me.Label8.Location = New System.Drawing.Point(272, 136)
Me.Label8.Name = "Label8"
Me.Label8.TabIndex = 18
Me.Label8.Text = "Issue Bank"
'
'Label7
'
Me.Label7.Location = New System.Drawing.Point(16, 136)
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(112, 23)
Me.Label7.TabIndex = 17
Me.Label7.Text = "Notirization Bank"
'
'Label6
'
Me.Label6.Location = New System.Drawing.Point(264, 96)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(112, 23)
```

```
Me.Label6.TabIndex = 16
Me.Label6.Text = "Negotiation Bank"
'
'Label5
'
Me.Label5.Location = New System.Drawing.Point(56, 96)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(64, 23)
Me.Label5.TabIndex = 15
Me.Label5.Text = "Pay Bank"
'
'Label4
'
Me.Label4.Location = New System.Drawing.Point(272, 64)
Me.Label4.Name = "Label4"
Me.Label4.TabIndex = 14
Me.Label4.Text = "Contract Number"
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(72, 64)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(48, 23)
Me.Label3.TabIndex = 13
Me.Label3.Text = "Amount"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(272, 32)
Me.Label2.Name = "Label2"
Me.Label2.TabIndex = 12
Me.Label2.Text = "Account Number"
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(16, 32)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(112, 23)
Me.Label1.TabIndex = 11
Me.Label1.Text = "Reference Number"
'
'ExpireDate
'
Me.ExpireDate.Location = New System.Drawing.Point(392, 176)
```

```
Me.ExpireDate.Name = "ExpireDate"
Me.ExpireDate.TabIndex = 10
Me.ExpireDate.Text = ""
'
'IssueBox
'
Me.IssueBox.Location = New System.Drawing.Point(392, 136)
Me.IssueBox.Name = "IssueBox"
Me.IssueBox.TabIndex = 9
Me.IssueBox.Text = ""
'
'NotBox
'
Me.NotBox.Location = New System.Drawing.Point(144, 136)
Me.NotBox.Name = "NotBox"
Me.NotBox.TabIndex = 8
Me.NotBox.Text = ""
'
'NegBox
'
Me.NegBox.Location = New System.Drawing.Point(392, 96)
Me.NegBox.Name = "NegBox"
Me.NegBox.TabIndex = 7
Me.NegBox.Text = ""
'
'PayBox
'
Me.PayBox.Location = New System.Drawing.Point(144, 96)
Me.PayBox.Name = "PayBox"
Me.PayBox.TabIndex = 6
Me.PayBox.Text = ""
'
'ContractBox
'
Me.ContractBox.Location = New System.Drawing.Point(392, 64)
Me.ContractBox.Name = "ContractBox"
Me.ContractBox.TabIndex = 5
Me.ContractBox.Text = ""
'
'AmountBox
'
Me.AmountBox.Location = New System.Drawing.Point(144, 64)
Me.AmountBox.Name = "AmountBox"
Me.AmountBox.ReadOnly = True
```

```
        Me.AmountBox.TabIndex = 4
        Me.AmountBox.Text = ""
        '
        'AcctBox
        '
        Me.AcctBox.Location = New System.Drawing.Point(392, 32)
        Me.AcctBox.Name = "AcctBox"
        Me.AcctBox.ReadOnly = True
        Me.AcctBox.TabIndex = 2
        Me.AcctBox.Text = ""
        '
        'RefBox
        '
        Me.RefBox.Location = New System.Drawing.Point(144, 32)
        Me.RefBox.Name = "RefBox"
        Me.RefBox.ReadOnly = True
        Me.RefBox.TabIndex = 0
        Me.RefBox.Text = ""
        '
        'ApplyButton
        '
        Me.ApplyButton.Location = New System.Drawing.Point(168, 328)
        Me.ApplyButton.Name = "ApplyButton"
        Me.ApplyButton.TabIndex = 1
        Me.ApplyButton.Text = "Apply"
        '
        'Form2
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
        Me.ClientSize = New System.Drawing.Size(616, 406)
        Me.Controls.Add(Me.GroupBox1)
        Me.Name = "Form2"
        Me.Text = "L/C Information"
        Me.GroupBox1.ResumeLayout(False)
        Me.ResultPanel.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region
    Public Structure LCinfo
        Public Ref_No As String
        Public LC_No As String
        Public Contract_No As String
```

```
        Public Issue_Date As Date
        Public Expire_Date As Date
        Public Not_Bank As String
        Public Neg_Bank As String
        Public Pay_Bank As String
        Public Amount As String
        Public Acct_No As String
        Public Memo As String

    End Structure

    Private Sub ExpireCalendar_DateChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.DateRangeEventArgs) Handles ExpireCalendar.DateChanged
        ExpireDate.Text = ExpireCalendar.SelectionStart

    End Sub

    Private Agent2 As New localhost2.Issue

    Private Sub ApplyButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ApplyButton.Click
        Dim lc As LCinfo
        Dim result As Boolean

        lc.LC_No = RefBox.Text + AmountBox.Text + Date.Today.ToShortDateString
        lc.Ref_No = RefBox.Text
        lc.Acct_No = AcctBox.Text
        lc.Amount = AmountBox.Text
        lc.Contract_No = ContractBox.Text
        lc.Expire_Date = Date.Parse(ExpireDate.Text)
        lc.Issue_Date = Date.Today
        lc.Memo = MemoBox.Text
        lc.Neg_Bank = NegBox.Text
        lc.Not_Bank = NotBox.Text
        lc.Pay_Bank = PayBox.Text

        result = Agent2.IssueLC(lc.LC_No, lc.Contract_No, lc.Ref_No, lc.Acct_No, lc.Issue_Date,
lc.Expire_Date, _
                                lc.Memo, lc.Not_Bank, lc.Neg_Bank, lc.Pay_Bank, lc.Amount)
        If result Then
            ExpireCalendar.Visible = False
            ResultPanel.Visible = True
            LCNoBox.Text = lc.LC_No
        Else
```

```
            MsgBox("Error")
        End If
        ApplyButton.Visible = False
    End Sub
End Class
```

VB code of AppLC Web service

```
Imports System.Web.Services
Imports System.Data
Imports System.Data.SqlClient

<WebService(Namespace:="http://tempuri.org/")> _
Public Class Apply
    Inherits System.Web.Services.WebService

#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
```

```
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    Public Structure Result
        Public str As String
        Public boo As Boolean
        Public i As Decimal
    End Structure

    <WebMethod(Description:="Method to apply a letter of credit.")> _
     Public Function ApplyLC(ByVal RefNo As String, ByVal Amount As Long, ByVal AcctNo As String) As
Result
        Dim MyConnection As SqlConnection
        Dim MyCommand As SqlDataAdapter
        Dim dataReader As SqlDataReader
        Dim search As String
        Dim am As Decimal
        Dim bal As Decimal
        Dim result As Result
        Dim rate As Decimal
        Dim Irate As Decimal

        Try
            search = "select Rate from tblClient where Ref_No='" & RefNo & "'"
            MyConnection = New SqlConnection("server = localhost; uid=sa;pwd=;
database=InternationalTrade")
            ' connect to database
            MyCommand = New SqlDataAdapter(search, MyConnection)
            MyConnection.Open()
            dataReader = MyCommand.SelectCommand.ExecuteReader()

            If dataReader.Read() Then
                rate = dataReader.GetValue(0)
                dataReader.Close()
            Else
                result.str = "Wrong Reference Number,please try again."
                result.boo = False
            End If

            search = "select Balance from tblAcct where Acct_No='" & AcctNo & "'and Ref_No= '" & RefNo
& "'"
```

```
MyCommand = New SqlDataAdapter(search, MyConnection)
dataReader = MyCommand.SelectCommand.ExecuteReader()
If dataReader.Read() Then
    bal = dataReader.GetValue(0)
    dataReader.Close()
    If rate >= 8 Then
        search = "select Percentage from tblRate where    Rate = '8'"
        MyCommand = New SqlDataAdapter(search, MyConnection)
        dataReader = MyCommand.SelectCommand.ExecuteReader()
        dataReader.Read()
        am = Amount * dataReader.GetValue(0)
        dataReader.Close()
    Else
        If rate >= 7 Then
            search = "select Percentage from tblRate where    Rate = '7'"
            MyCommand = New SqlDataAdapter(search, MyConnection)
            dataReader = MyCommand.SelectCommand.ExecuteReader()
            dataReader.Read()
            am = Amount * dataReader.GetValue(0)
            dataReader.Close()
        Else
            If rate >= 6 Then
                search = "select Percentage from tblRate where    Rate = '6'"
                MyCommand = New SqlDataAdapter(search, MyConnection)
                dataReader = MyCommand.SelectCommand.ExecuteReader()
                dataReader.Read()
                am = Amount * dataReader.GetValue(0)
                dataReader.Close()
            Else
                If rate >= 5 Then
                    search = "select Percentage from tblRate where    Rate = '5'"
                    MyCommand = New SqlDataAdapter(search, MyConnection)
                    dataReader = MyCommand.SelectCommand.ExecuteReader()
                    dataReader.Read()
                    am = Amount * dataReader.GetValue(0)
                    dataReader.Close()
                Else
                    result.str = "Not reliable client"
                    result.boo = False
                End If
            End If

        End If
    End If
```

```
                Irate = 0.035 * 100
                If bal >= am Then
                    result.str = "You are qualified to our prerequisite.Depend on your situation, we require to
hold $" + am.ToString + " as " & _
                                "your guarantee deposit. Your intereste rate is " + Irate.ToString + "% ." &
_
                                "You can continue to apply your letter of Credit now."
                    result.i = Irate / 100
                    result.boo = True
                Else
                    Irate = Irate + 1.5
                    result.str = "You are not qualified to our prerequisite. Because the balance in your account
is lower than" & _
                                " our requirment. Depands on your case, we can issue your letter of credit with
interest rate = primary rate +" & _
                                " 1.5% = " + Irate.ToString + "% . If you agree with it, you can continue to
apply your letter of credit now. Otherwise, please" & _
                                " click close button to terminate your application."
                    result.i = Irate / 100
                    result.boo = False
                End If
            Else
                result.str = "Wrong Account Number,please try again."
                result.boo = False
                result.i = 0
            End If

        Catch exception As SqlException
            result.str = "Error sqlconnection"
            result.boo = False
            result.i = 0
        Finally
            MyConnection.Close()
        End Try
        Return result
    End Function
End Class
```

B. VB code of IssueLC Web service

```
Imports System.Web.Services
Imports System.Data
```

```
Imports System.Data.SqlClient

<System.Web.Services.WebService(Namespace:="http://tempuri.org/LCIssue/Service1")> _
Public Class Issue
    Inherits WebService
    'Inherits System.Web.Services.WebService

#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    <WebMethod(Description:="Method to issue a letter of credit.")> _
        Public Function IssueLC(ByVal LC_No As String, ByVal Contract_No As String, ByVal Ref_No As
```

```
String, _
                ByVal Acct_No As String, ByVal Issue_Date As Date, ByVal Expire_Date As Date, ByVal
Memo As String, _
                ByVal Not_Bank As String, ByVal Neg_Bank As String, ByVal Pay_Bank As String, ByVal
Amount As String) As Boolean

        Dim MyConnection As SqlConnection
        Dim MyCommand As SqlDataAdapter
        Dim dataReader As SqlDataReader
        Dim insertcmd As String
        Dim cmd As SqlCommand
        Dim message As String

        message = "begin"
        MyConnection = New SqlConnection("server = localhost; uid=sa;pwd=; database=InternationalTrade")
        MyConnection.Open()
        'connect to database
        insertcmd = "insert into tblLC values
(@LC_No,@Contract_No,@Ref_No,@Acct_No,@Issue_Date,@Expire_Date,@Memo," & _
          "@Not_Bank,@Neg_Bank,@Pay_Bank,@Amount);"
        cmd = New SqlCommand(insertcmd, MyConnection)
        cmd.Parameters.Add(New SqlParameter("@LC_No", SqlDbType.Char, 30))
        cmd.Parameters("@LC_No").Value = LC_No
        cmd.Parameters.Add(New SqlParameter("@Contract_No", SqlDbType.Char, 10))
        cmd.Parameters("@Contract_No").Value = Contract_No
        cmd.Parameters.Add(New SqlParameter("@Ref_No", SqlDbType.Char, 10))
        cmd.Parameters("@Ref_No").Value = Ref_No
        cmd.Parameters.Add(New SqlParameter("@Acct_No", SqlDbType.Char, 10))
        cmd.Parameters("@Acct_No").Value = Acct_No
        cmd.Parameters.Add(New SqlParameter("@Issue_Date", SqlDbType.DateTime, 8))
        cmd.Parameters("@Issue_Date").Value = Date.Parse(Issue_Date)
        cmd.Parameters.Add(New SqlParameter("@Expire_Date", SqlDbType.DateTime, 8))
        cmd.Parameters("@Expire_Date").Value = Expire_Date
        cmd.Parameters.Add(New SqlParameter("@Memo", SqlDbType.Char, 500))
        cmd.Parameters("@Memo").Value = Memo
        cmd.Parameters.Add(New SqlParameter("@Not_Bank", SqlDbType.Char, 50))
        cmd.Parameters("@Not_Bank").Value = Not_Bank
        cmd.Parameters.Add(New SqlParameter("@Neg_Bank", SqlDbType.Char, 50))
        cmd.Parameters("@Neg_Bank").Value = Neg_Bank
        cmd.Parameters.Add(New SqlParameter("@Pay_Bank", SqlDbType.Char, 50))
        cmd.Parameters("@Pay_Bank").Value = Pay_Bank
        cmd.Parameters.Add(New SqlParameter("@Amount", SqlDbType.Char, 10))
        cmd.Parameters("@Amount").Value = Amount
        ' insert a new record
```

```
        Try
            cmd.ExecuteNonQuery()
        Catch ex As SqlException
            message = "error in sql execute"
            Throw New Exception(message)
        End Try

        cmd.Connection.Close()

        If message = "begin" Then
            Return True
        Else
            Return False
        End If

    End Function
End Class
```

C. VB code of Insurance Interface Web service

```
Imports System.data
Imports System.Data.SqlClient
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/InsuranceInterface/Service1")> _
Public Class InsuranceInterface
    Inherits System.Web.Services.WebService

#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer
    Private Agent1 As New localhost1.Quote
```

```
    Private Agent2 As New localhost2.Quote
    Private Agent3 As New localhost3.Quote

    'Private Agent As New localhost2
    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        components = New System.ComponentModel.Container
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    Public Structure Quote
        Public name As String
        Public fee As String
    End Structure

    <WebMethod(Description:="Method to get insurance quotes.")> _
      Public Function InsuranceInterface(ByVal price As String, ByVal type As String) As Boolean
        Dim ds As DataSet
        Dim qt As Quote
        Dim conn As SqlConnection
        Dim cmd As SqlCommand
        Dim myadapter As SqlDataAdapter


        conn = New SqlConnection("server = localhost; uid=sa;pwd=; database=ERP")
        conn.Open()
        cmd = New SqlCommand("delete from tblInsurance", conn)
        cmd.ExecuteNonQuery()

        qt.name = Agent1.Quotes(type, price).company
```

```
        qt.fee = Agent1.Quotes(type, price).fee
        InsertQuote(qt)

        qt.name = Agent2.Quotes(type, price).company
        qt.fee = Agent2.Quotes(type, price).fee

        InsertQuote(qt)

        qt.name = Agent3.Quotes(type, price).company
        qt.fee = Agent3.Quotes(type, price).fee
        InsertQuote(qt)
        Return True
    End Function

    Private Sub InsertQuote(ByVal qt As Quote)
        Dim conn As SqlConnection
        Dim cmd As SqlCommand
        Dim insertcmd As String
        Dim ds As DataSet
        Dim message As String

        message = "begin"
        conn = New SqlConnection("server = localhost; uid=sa;pwd=; database=ERP")
        conn.Open()
        'connect to database
        insertcmd = "insert into tblInsurance values (@CompanyName,@Fee);"
        cmd = New SqlCommand(insertcmd, conn)
        cmd.Parameters.Add(New SqlParameter("@CompanyName", SqlDbType.Char, 50))
        cmd.Parameters("@CompanyName").Value = qt.name
        cmd.Parameters.Add(New SqlParameter("@Fee", SqlDbType.Real, 10))
        cmd.Parameters("@Fee").Value = qt.fee

        cmd.ExecuteNonQuery()
        conn.Close()
    End Sub
End Class
```

D. VB code of Insurance Quote Web service

```
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/QuotesWS/Quote")> _
Public Class Quote
    Inherits System.Web.Services.WebService
```

```
#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        components = New System.ComponentModel.Container
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    Public Structure Quote
        Dim fee As Single
        Dim company As String
    End Structure

    <WebMethod(Description:="Method to quote a shipping insurance.")> _
            Public Function Quotes(ByVal ShipType As String, ByVal PriceofGoods As String) As Quote

        Dim type As String
```

```
        Dim quote As Quote

        Select Case ShipType
            Case "Air"
                quote.fee = 250 + PriceofGoods * 0.015
            Case "Sea"
                quote.fee = 800 + PriceofGoods * 0.01
            Case "Land"
                quote.fee = 300 + PriceofGoods * 0.0075
        End Select
        quote.company = "ING Insurance Company"
        Return quote
    End Function
End Class
```

E. VB code of Shipping Interface Web service

```
Imports System.Web.Services

<System.Web.Services.WebService(Namespace := "http://tempuri.org/ShipInterfaceWS/Service1")> _
Public Class Service1
    Inherits System.Web.Services.WebService

#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer
    Private Agent1 As New localhost1.Shippment
    Private Agent2 As New localhost2.Shippment
    Private Agent3 As New localhost3.Shippment

    'Private Agent As New localhost2
    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
```

```
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        components = New System.ComponentModel.Container()
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    Public Structure Quote
        Public name As String
        Public price As String
        Public result As String

    End Structure

    <WebMethod(Description:="Method to get shipping quotes.")> _
      Public Function ShippInterface(ByVal quantity As String, ByVal type As String, ByVal size As Short) As
Boolean
        Dim ds As DataSet
        Dim qt As Quote
        Dim conn As SqlConnection
        Dim cmd As SqlCommand
        Dim myadapter As SqlDataAdapter

        conn = New SqlConnection("server = localhost; uid=sa;pwd=; database=ERP")
        conn.Open()
        cmd = New SqlCommand("delete from tblQuotes", conn)
        cmd.ExecuteNonQuery()

        qt.name = Agent1.Shipping(type, size, quantity).name
        qt.price = Agent1.Shipping(type, size, quantity).price
        qt.result = Agent1.Shipping(type, size, quantity).result
        InsertQuote(qt)

        qt.name = Agent2.Shipping(type, size, quantity).name
```

```
        qt.price = Agent2.Shipping(type, size, quantity).price
        qt.result = Agent2.Shipping(type, size, quantity).result
        InsertQuote(qt)

        qt.name = Agent3.Shipping(type, size, quantity).name
        qt.price = Agent3.Shipping(type, size, quantity).price
        qt.result = Agent3.Shipping(type, size, quantity).result
        InsertQuote(qt)
        Return True
    End Function

    Private Sub InsertQuote(ByVal qt As Quote)
        Dim conn As SqlConnection
        Dim cmd As SqlCommand
        Dim insertcmd As String
        Dim ds As DataSet
        Dim message As String

        message = "begin"
        conn = New SqlConnection("server = localhost; uid=sa;pwd=; database=ERP")
        conn.Open()
        'connect to database
        insertcmd = "insert into tblQuotes values (@name,@price,@result);"
        cmd = New SqlCommand(insertcmd, conn)
        cmd.Parameters.Add(New SqlParameter("@name", SqlDbType.Char, 50))
        cmd.Parameters("@name").Value = qt.name
        cmd.Parameters.Add(New SqlParameter("@price", SqlDbType.Real, 10))
        cmd.Parameters("@price").Value = qt.price
        cmd.Parameters.Add(New SqlParameter("@result", SqlDbType.Char, 50))
        cmd.Parameters("@result").Value = qt.result

        cmd.ExecuteNonQuery()
        conn.Close()
    End Sub

End Class
```

F. VB code of Shipping Quote Web service

```
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/ShippingWS/Shippment")> _
Public Class Shippment
    Inherits System.Web.Services.WebService
```

```
#Region " Web services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web services Designer
    'It can be modified using the Web services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        components = New System.ComponentModel.Container
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region

    Public Structure Quote
        Public name As String
        Public price As Double
        Public result As String
    End Structure

    <WebMethod(Description:="Method to get a shipping rate.")> _
            Public Function Shipping(ByVal ShipType As String, ByVal Size As String, ByVal Quantity As
String) As Quote
```

```
        Dim rate As Single
        Dim s As String
        Dim quote As Quote

        quote.name = "Mcdonald Shipping Company"
        quote.result = "True"
        Select Case ShipType
            Case "Air"
                If Size > 1000 Then
                    quote.price = 0.0
                    quote.result = "Your size is over our capability! Please try other shipping type!"
                Else
                    quote.price = 500 + Size * 0.01 + Quantity * 0.0225
                    'quote.price = quote.price + Distance * 0.01
                End If
            Case "Sea"
                quote.price = 800 + Size * 0.02 + Quantity * 0.035
                'quote.price = quote.price + Distance * 0.01
            Case "Land"
                quote.price = 200 + Size * 0.02 + Quantity * 0.02
                'quote.price = quote.price + Distance * 0.01
        End Select
        Return quote

    End Function
End Class
```

G. SOAP of AppLC Web service

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /LCApp/ApplyLC.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ApplyLC"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ApplyLC xmlns="http://tempuri.org/">
```

```
      <RefNo>string</RefNo>
      <Amount>long</Amount>
      <AcctNo>string</AcctNo>
    </ApplyLC>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ApplyLCResponse xmlns="http://tempuri.org/">
      <ApplyLCResult>
        <str>string</str>
        <boo>boolean</boo>
        <i>decimal</i>
      </ApplyLCResult>
    </ApplyLCResponse>
  </soap:Body>
</soap:Envelope>
```

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /LCApp/ApplyLC.asmx/ApplyLC HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

RefNo=string&Amount=string&AcctNo=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<Result xmlns="http://tempuri.org/">
  <str>string</str>
  <boo>boolean</boo>
  <i>decimal</i>
</Result>
```

H. WSDL file of AppLC WSDL

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
- <s:element name="ApplyLC">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="RefNo" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="Amount" type="s:long" />
  <s:element minOccurs="0" maxOccurs="1" name="AcctNo" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="ApplyLCResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="ApplyLCResult" type="s0:Result" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:complexType name="Result">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="str" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="boo" type="s:boolean" />
  <s:element minOccurs="1" maxOccurs="1" name="i" type="s:decimal" />
  </s:sequence>
  </s:complexType>
  </s:schema>
  </types>
- <message name="ApplyLCSoapIn">
  <part name="parameters" element="s0:ApplyLC" />
  </message>
- <message name="ApplyLCSoapOut">
  <part name="parameters" element="s0:ApplyLCResponse" />
  </message>
- <portType name="ApplySoap">
```

```
- <operation name="ApplyLC">
  <documentation>Method to apply a letter of credit.</documentation>
  <input message="s0:ApplyLCSoapIn" />
  <output message="s0:ApplyLCSoapOut" />
  </operation>
  </portType>
- <binding name="ApplySoap" type="s0:ApplySoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="ApplyLC">
  <soap:operation soapAction="http://tempuri.org/ApplyLC" style="document" />
- <input>
  <soap:body use="literal" />
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Apply">
- <port name="ApplySoap" binding="s0:ApplySoap">
  <soap:address location="http://localhost/LCApp/ApplyLC.asmx" />
  </port>
  </service>
  </definitions>
```

I. SOAP file of IssueLC

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /LCIssue/IssueLC.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/LCIssue/Service1/IssueLC"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <IssueLC xmlns="http://tempuri.org/LCIssue/Service1">
      <LC_No>string</LC_No>
      <Contract_No>string</Contract_No>
      <Ref_No>string</Ref_No>
```

```
        <Acct_No>string</Acct_No>
        <Issue_Date>dateTime</Issue_Date>
        <Expire_Date>dateTime</Expire_Date>
        <Memo>string</Memo>
        <Not_Bank>string</Not_Bank>
        <Neg_Bank>string</Neg_Bank>
        <Pay_Bank>string</Pay_Bank>
        <Amount>string</Amount>
      </IssueLC>
    </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <IssueLCResponse xmlns="http://tempuri.org/LCIssue/Service1">
      <IssueLCResult>boolean</IssueLCResult>
    </IssueLCResponse>
  </soap:Body>
</soap:Envelope>
HTTP POST
The following is a sample HTTP POST request and response. The placeholders shown need to be
replaced with actual values.
POST /LCIssue/IssueLC.asmx/IssueLC HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

LC_No=string&Contract_No=string&Ref_No=string&Acct_No=string&Issue_Date=string&Expire_D
ate=string&Memo=string&Not_Bank=string&Neg_Bank=string&Pay_Bank=string&Amount=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<boolean xmlns="http://tempuri.org/LCIssue/Service1">boolean</boolean>
```

J. WSDL file of IssueLC

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/LCIssue/Service1"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/LCIssue/Service1" xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/LCIssue/Service1">
- <s:element name="IssueLC">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="LC_No" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Contract_No" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Ref_No" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Acct_No" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="Issue_Date" type="s:dateTime" />
  <s:element minOccurs="1" maxOccurs="1" name="Expire_Date" type="s:dateTime" />
  <s:element minOccurs="0" maxOccurs="1" name="Memo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Not_Bank" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Neg_Bank" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Pay_Bank" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Amount" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="IssueLCResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="IssueLCResult" type="s:boolean" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </types>
- <message name="IssueLCSoapIn">
  <part name="parameters" element="s0:IssueLC" />
  </message>
- <message name="IssueLCSoapOut">
  <part name="parameters" element="s0:IssueLCResponse" />
  </message>
- <portType name="IssueSoap">
```

```
- <operation name="IssueLC">
  <documentation>Method to issue a letter of credit.</documentation>
  <input message="s0:IssueLCSoapIn" />
  <output message="s0:IssueLCSoapOut" />
  </operation>
  </portType>
- <binding name="IssueSoap" type="s0:IssueSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="IssueLC">
  <soap:operation soapAction="http://tempuri.org/LCIssue/Service1/IssueLC" style="document" />
- <input>
  <soap:body use="literal" />
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Issue">
- <port name="IssueSoap" binding="s0:IssueSoap">
  <soap:address location="http://localhost/LCIssue/IssueLC.asmx" />
  </port>
  </service>
  </definitions>
```

K. SOAP file of ShippingInterface

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /ShipInterfaceWS/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ShipInterfaceWS/Service1/ShippInterface"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ShippInterface xmlns="http://tempuri.org/ShipInterfaceWS/Service1">
      <quantity>string</quantity>
      <type>string</type>
      <size>short</size>
```

```
    </ShippInterface>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ShippInterfaceResponse xmlns="http://tempuri.org/ShipInterfaceWS/Service1">
      <ShippInterfaceResult>boolean</ShippInterfaceResult>
    </ShippInterfaceResponse>
  </soap:Body>
</soap:Envelope>
HTTP POST
The following is a sample HTTP POST request and response. The placeholders shown need to be
replaced with actual values.
POST /ShipInterfaceWS/Service1.asmx/ShippInterface HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

quantity=string&type=string&size=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<boolean xmlns="http://tempuri.org/ShipInterfaceWS/Service1">boolean</boolean>
```

L. WSDL file of ShippingInterface

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://tempuri.org/ShipInterfaceWS/Service1"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
```

```
targetNamespace="http://tempuri.org/ShipInterfaceWS/Service1"
xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/ShipInterfaceWS/Service1">
- <s:element name="ShippInterface">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="quantity" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="type" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="size" type="s:short" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="ShippInterfaceResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="ShippInterfaceResult" type="s:boolean" />
  </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </types>
- <message name="ShippInterfaceSoapIn">
  <part name="parameters" element="s0:ShippInterface" />
  </message>
- <message name="ShippInterfaceSoapOut">
  <part name="parameters" element="s0:ShippInterfaceResponse" />
  </message>
- <portType name="Service1Soap">
- <operation name="ShippInterface">
  <documentation>Method to get shipping quotes.</documentation>
  <input message="s0:ShippInterfaceSoapIn" />
  <output message="s0:ShippInterfaceSoapOut" />
  </operation>
  </portType>
- <binding name="Service1Soap" type="s0:Service1Soap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="ShippInterface">
  <soap:operation soapAction="http://tempuri.org/ShipInterfaceWS/Service1/ShippInterface"
style="document" />
- <input>
  <soap:body use="literal" />
  </input>
```

```
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Service1">
- <port name="Service1Soap" binding="s0:Service1Soap">
  <soap:address location="http://localhost/ShipInterfaceWS/Service1.asmx" />
  </port>
  </service>
  </definitions>
```

M. SOAP file of ShippingQuote

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /ShippingWS/Shippment.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ShippingWS/Shippment/Shipping"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Shipping xmlns="http://tempuri.org/ShippingWS/Shippment">
      <ShipType>string</ShipType>
      <Size>string</Size>
      <Quantity>string</Quantity>
    </Shipping>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```
    <ShippingResponse xmlns="http://tempuri.org/ShippingWS/Shippment">
      <ShippingResult>
        <name>string</name>
        <price>double</price>
        <result>string</result>
      </ShippingResult>
    </ShippingResponse>
  </soap:Body>
</soap:Envelope>
```

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /ShippingWS/Shippment.asmx/Shipping HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

ShipType=string&Size=string&Quantity=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<Quote xmlns="http://tempuri.org/ShippingWS/Shippment">
  <name>string</name>
  <price>double</price>
  <result>string</result>
</Quote>
```

N. WSDL file of ShippingQuote

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://tempuri.org/ShippingWS/Shippment"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/ShippingWS/Shippment"
xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
```

```
- <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/ShippingWS/Shippment">
- <s:element name="Shipping">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="ShipType" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Size" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Quantity" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="ShippingResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="ShippingResult" type="s0:Quote" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:complexType name="Quote">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="price" type="s:double" />
  <s:element minOccurs="0" maxOccurs="1" name="result" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:schema>
  </types>
- <message name="ShippingSoapIn">
  <part name="parameters" element="s0:Shipping" />
  </message>
- <message name="ShippingSoapOut">
  <part name="parameters" element="s0:ShippingResponse" />
  </message>
- <portType name="ShippmentSoap">
- <operation name="Shipping">
  <documentation>Method to get a shipping rate.</documentation>
  <input message="s0:ShippingSoapIn" />
  <output message="s0:ShippingSoapOut" />
  </operation>
  </portType>
- <binding name="ShippmentSoap" type="s0:ShippmentSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="Shipping">
```

```
  <soap:operation soapAction="http://tempuri.org/ShippingWS/Shippment/Shipping"
style="document" />
- <input>
  <soap:body use="literal" />
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Shippment">
- <port name="ShippmentSoap" binding="s0:ShippmentSoap">
  <soap:address location="http://localhost/ShippingWS/Shippment.asmx" />
  </port>
  </service>
  </definitions>
```

O. SOAP file of InsuranceInterface

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /InsuranceInterface/InsuranceInterface.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/InsuranceInterface/Service1/InsuranceInterface"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsuranceInterface xmlns="http://tempuri.org/InsuranceInterface/Service1">
      <price>string</price>
      <type>string</type>
    </InsuranceInterface>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InsuranceInterfaceResponse xmlns="http://tempuri.org/InsuranceInterface/Service1">
      <InsuranceInterfaceResult>boolean</InsuranceInterfaceResult>
    </InsuranceInterfaceResponse>
  </soap:Body>
</soap:Envelope>
```

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /InsuranceInterface/InsuranceInterface.asmx/InsuranceInterface HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

price=string&type=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<boolean xmlns="http://tempuri.org/InsuranceInterface/Service1">boolean</boolean>
```

P. WSDL file of InsuranceInterface

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://tempuri.org/InsuranceInterface/Service1"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/InsuranceInterface/Service1"
xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/InsuranceInterface/Service1">
- <s:element name="InsuranceInterface">
- <s:complexType>
- <s:sequence>
```

```
<s:element minOccurs="0" maxOccurs="1" name="price" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="type" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="InsuranceInterfaceResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="InsuranceInterfaceResult" type="s:boolean" />
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</types>
- <message name="InsuranceInterfaceSoapIn">
<part name="parameters" element="s0:InsuranceInterface" />
</message>
- <message name="InsuranceInterfaceSoapOut">
<part name="parameters" element="s0:InsuranceInterfaceResponse" />
</message>
- <portType name="InsuranceInterfaceSoap">
- <operation name="InsuranceInterface">
<documentation>Method to get insurance quotes.</documentation>
<input message="s0:InsuranceInterfaceSoapIn" />
<output message="s0:InsuranceInterfaceSoapOut" />
</operation>
</portType>
- <binding name="InsuranceInterfaceSoap" type="s0:InsuranceInterfaceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="InsuranceInterface">
<soap:operation soapAction="http://tempuri.org/InsuranceInterface/Service1/InsuranceInterface"
style="document" />
- <input>
<soap:body use="literal" />
</input>
- <output>
<soap:body use="literal" />
</output>
</operation>
</binding>
- <service name="InsuranceInterface">
- <port name="InsuranceInterfaceSoap" binding="s0:InsuranceInterfaceSoap">
<soap:address location="http://localhost/InsuranceInterface/InsuranceInterface.asmx" />
</port>
```

```
</service>
</definitions>
```

Q. SOAP file of InsuranceQuote

The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.

```
POST /Insurance3/Quote3.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/QuotesWS/Quote/Quotes"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Quotes xmlns="http://tempuri.org/QuotesWS/Quote">
      <ShipType>string</ShipType>
      <PriceofGoods>string</PriceofGoods>
    </Quotes>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <QuotesResponse xmlns="http://tempuri.org/QuotesWS/Quote">
      <QuotesResult>
        <fee>float</fee>
        <company>string</company>
      </QuotesResult>
    </QuotesResponse>
  </soap:Body>
</soap:Envelope>
HTTP POST
```

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /Insurance3/Quote3.asmx/Quotes HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

ShipType=string&PriceofGoods=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<Quote xmlns="http://tempuri.org/QuotesWS/Quote">
  <fee>float</fee>
  <company>string</company>
</Quote>
```

R. WSDL file of InsuranceQuote

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/QuotesWS/Quote"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://tempuri.org/QuotesWS/Quote" xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/QuotesWS/Quote">
- <s:element name="Quotes">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="ShipType" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="PriceofGoods" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="QuotesResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="QuotesResult" type="s0:Quote" />
```

```
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:complexType name="Quote">
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="fee" type="s:float" />
  <s:element minOccurs="0" maxOccurs="1" name="company" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:schema>
  </types>
- <message name="QuotesSoapIn">
  <part name="parameters" element="s0:Quotes" />
  </message>
- <message name="QuotesSoapOut">
  <part name="parameters" element="s0:QuotesResponse" />
  </message>
- <portType name="QuoteSoap">
- <operation name="Quotes">
  <documentation>Method to quote a shipping insurance.</documentation>
  <input message="s0:QuotesSoapIn" />
  <output message="s0:QuotesSoapOut" />
  </operation>
  </portType>
- <binding name="QuoteSoap" type="s0:QuoteSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="Quotes">
  <soap:operation soapAction="http://tempuri.org/QuotesWS/Quote/Quotes" style="document" />
- <input>
  <soap:body use="literal" />
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="Quote">
- <port name="QuoteSoap" binding="s0:QuoteSoap">
  <soap:address location="http://localhost/Insurance3/Quote3.asmx" />
  </port>
  </service>
  </definitions>
```

Reference

1 Web services Architecture RequirementsW3C Working Draft 14 November 2002 http://www.w3.org/TR/wsa-reqs#id260483, accessed by November,2004

2 Java Web services Programming by Rashim Mogha and V.V. Preetham, 2002

3 J. Borck, "InfoWorld Technology of the Year: Web services," InfoWorld 4 February 2002: 48

4 W.Wong, "Why Web services Make Business Sense," http://news.com.com/2009-1017-275442.html?legacy=cnet, accessed by December,2004

5 T.Olavsrud, "Microsoft Puts.NET My Services on Hold," 11 April 2002 www.internetnews.com/dev-news/article/0,,10-1007961,00.html, accessed by August,2004

6 "The XML Cover Pages: IBM Global Services and IBM WebSphere Platform to support IBM's Web services Infrastructure," 22 May 2001 http://xml.coverpages.org/ni2001-05-22-b.html, accessed by January,2005

7 "Enterprise Developer for Multiplatforms," http://www.ibm.com/redpapers/padfs/redp0414.pdf , accessed by September,2004

8 T.Kontzer and A.Gonsalves. "Lotus Embraces Web services, But Will Customers?," 4 February 2002 www.informationweek.com/sotry/IWK20020201s0029, accessed by January,2005

9 K.Ohlson, "Big Blue to Trumpet Web services Scheme," Network World 14 May 2001:68

10 SUN Open Net Environment (SunONE), http://www.sun.com/software/sunone, accessed by January,2005

11 R.Adhikari, "Sun Extends Web services Strategy," Application Development Trends December 2001:12

12 "SUN ONE Overview: vision," http:// www.sun.com/software/sunone/overview/vision, accessed by November,2004

13 C.Goldfarb and P.Prescod, The XML Handbook, Third Edition. Upper Saddle River, New Jersey: Prentice Hall, 2001:20

14 C.Goldfarb and P.Prescod, The XML Handbook, Third Edition. Upper Saddle River, New Jersey: Prentice Hall, 2001: 19

15 R. Boeri, "XML Across the Publishing Lifecycle," eContent October 2001:21

16 "SGML Users' Group History," http://www.oaisi-open.org/cover/sgmilhisto.html, accessed by June,2004

17 C.Goldfarb and P.Prescod, The XML Handbook, Third Edition. Upper Saddle River, New Jersey: Prentice Hall, 2001:21

18 J.Norton, "XML Fundamentals," DB2 Magazine WUarter2,2001:52

19 Integrating core business functions, http://www-306.ibm.com/software/ebusiness/jstart/casestudies/cibc.shtml, accessed by

November,2004

20 Integrating core business functions,
http://www-306.ibm.com/software/ebusiness/jstart/casestudies/cibc.shtml, accessed by November,2004

21 Integrating core business functions,
http://www-306.ibm.com/software/ebusiness/jstart/casestudies/cibc.shtml, accessed by November,2004

22 XML Web services in the Financial Services Industry: Will we ever cross the firewall?
http://www.celent.com/PressReleases/20040405/WebServices.htm, accessed by November,2004

23 "About us", http://www.standardandpoors.com, accessed by November,2004

24 Kimberly B. Caisse, "S&P Turns Java Beans into Web services," March 15, 2002
http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2855469,00.html, accessed by November,2004

25 Richard Kappinski, "S&P's Web services Play," InternetWeek November 1, 2001,
http://www.internetweek.com/transtoday01/ttoday110101.htm, accessed by October,2004

26 Web services and Straight Through Processing, Gunjan Samtani and Dimple Sadhwani,
http://www.webservicesarchitect.com/content/articles/samtani06.asp, accessed by November,2004

27 Message and processing quality - the backbone of payments STP solutions, 14 August 2003
http://www.swift.com/index.cfm?item_id=42498, accessed by October,2004

28 Message and processing quality - the backbone of payments STP solutions, Lambert Timmermans, Business Manager, Global Initiatives & STP, SWIFT,
http://www.swift.com/index.cfm?item_id=42498, accessed by October,2004

29 FpML sample, http://www.fpml.org/documents/proposals/valuation/val_ex01_rfqresp.xml, accessed by October,2004

30 FIX sample, http://www.fixprotocol.org/documents/611/ExampleFixMessages.txt, accessed by October,2004

31 Web services and Straight Through Processing, Gunjan Samtani and Dimple Sadhwani
http://www.webservicesarchitect.com/content/articles/samtani06.asp, Accessed by October,2003

32 Maria Trombly, Web services: A Dream Deferred,
http://securitiesindustry.com/article.cfm?articleid=393&searchTerm=trombly%20sapient, accessed by October,2004

33 Susan Landry, The Role of Web services in the Financial Services Industry,
http://www.gartnerg2.com/research/rpt-0702-0107.asp, accessed by October,2004

34 Mitali Kalita Web services in Capital Market,
http://www.investment-index.com/newsletters/issue11.html, accessed by nov,2004

35 Trade services need client-centric banks,
http://www.swift.com/index.cfm?item_id=5571#instruments, accessed by October, 2004

36 SWIFTStandars_tread.pdf, Aug, 2004, http://www.swift.com, accessed by November, 2004

37 SWIFTStandars_tread.pdf, Aug, 2004, http://www.swift.com, accessed by November, 2004

38 Simple XML, An introduction for the SWIFT community, Aug, 2004, http://www.swift.com ,accessed by November, 2004

39 SWIFTStandards XML for Implementors, January 2004, http://www.swift.com/index.cfm?item_id=41815, accessed by January,2005

40 Swift Launches XML-Based Payments Initiative :4 Oct 2002 http://lighthouse-partners.com/xml/newsarchive/20021004b.htm, accessed by November,2004

41 Geneva, Switzerland Cape Clear Announces Web services Integration for SWIFT Financial Network, September 30, 2002 http://xml.coverpages.org/CapeClear-SWIFT.html, accessed by November,2004

42 Annraí O'Toole, capeclear_swiftbrochure.pdf

43 SWIFTNet providing single window access for financial world, http://www.swift.com, accessed by November,2004

44 Cape Clear Software Brings SWIFT Web services to the Financial Community SAN MATEO, Calif. September 9, 2002 http://www.capeclear.com/news/press_releases/reports/swift.shtml, accessed by November,2004

45 Hall, A. D., (1969) Three Dimensional Morphology of Systems Engineering, IEEE Transactions on Systems, Man and Cybernetics, April, 1969

46 Quality of Service for Web services—Demystification, Limitations, and Best Practices By Rajesh Sumra and Arulazi D http://www.developer.com/services/article.php/10928_2027911_2