AN EXCHANGE SERVICE FOR FINANCIAL MARKETS

Hairong Yu, Fethi A. Rabhi and Feras T. Dabous
School of Information Systems, Technology and Management,The University of New South Wales, Sydney 2052, Australia

Keywords: Information Systems Integration, Legacy Systems, Web services, Performance, Security, Trading Systems

Abstract: The critical business requirements and compelling nature of the competitive landscape are pushing Information Technology systems away from the traditional centrally controlled corporate-wide architectures towards dynamic, loosely coupled, self-defining and service-based solutions. Web services are regarded as a key technology for addressing the need for connecting extended applications and providing standards and flexibility for enterprise legacy systems integration. This paper reports our experiences when integrating a financial market trading system. This integration process starts from analysing the trading system's architecture, then identifying system functionality and finally realising the design and implementation of a Web service. Performance and security and the trade-offs involved are the major focus points throughout this process. Comprehensive benchmarking is conducted with and without Web service and security considerations.

1 INTRODUCTION

Information technology innovations such as electronic communication networks have enormous impact on the way people are doing business nowadays. For example, the finance industry is evolving rapidly as companies adapt new technologies in order to survive in a highly competitive and dynamic business environment. The explosive growth in online trading systems that operate from different locations worldwide is just one example. However many mature, well developed existing legacy trading systems do not meet new business requirements. Their design prevents them from interacting with other financial systems in order to provide more services and extend enterprise solutions. Developing applications which try to supply best approaches to solve this problem is becoming highest priority (Hendershott, 2003).

As interoperable trading systems are becoming more common (Allen, 2001), this paper is concerned with adapting centrally-controlled corporate-wide legacy systems to satisfy these new requirements. There are many challenges:

- **Business Partner Integration:** a centrally controlled architecture hardly ensures complete, expeditious and flexible integration between systems including legacy systems when new business partners are introduced or supply chains cross a community are needed to be chosen or outsourcing units are required to be seamlessly connected.
- **Service Accessibility:** traditionally packaged and tightly coupled systems cannot easily be delivered as streams of services and accessed pervasively from anywhere.
- **System Development:** slow software development obstructs organizations' ability to expose system functionalities for external use on value added tasks or extend to new markets.
- **Standardization:** individual specifications and standardizations are carried out without industry agreements on data exchange, messaging, interface description and business process layer.

New technologies such as Web Services are pushed by many key IT vendors (IBM, Microsoft, Hewlett-Packard, Oracle, Sun etc) and do not require the adoption of a common implementation platform. The technologies only require conformity to some standard protocols and are extremely efficient for leveraging existing applications and infrastructure. Despite those promises (CBDI, 2003), there are not many Web services currently operating in important application domains, particularly financial trading.

One of the impediments is that in addition to integration difficulties, there are serious concerns about meeting industry essential requirements in performance and security.

To study the application of Web service technology in financial trading, we are developing a proof-of-concept prototype Web service called Exchange Service (Exchange Service, 2003) while addressing those critical requirements. The Exchange Service is based on a full-fledged commercial financial market trading system called X-Stream (Computershare, 2003) which has been developed for many years (Trading Technology, 2003).

This paper reports our experiences in developing the Exchange Service. It is organised as follows: section 2 gives background on trading systems and their industry requirements. Section 3 illustrates the Exchange Service design and implementation. Experiments involving two focal points, namely performance and security, are discussed in section 4. Conclusions and future work are presented in section 5.

2 BACKGROUND AND REQUIREMENTS

In this section, the application domain is briefly introduced. Future directions and industry principal requirements are also identified.

2.1 Financial Markets and Trading Systems

A financial market's purpose is to facilitate the exchange of financial assets (Viney, 2000). A trading system or a trading engine is a computerized system used to trade financial products such as equities, options, futures, currencies and commodities in financial markets (Harris, 2002). Business rules related to financial markets can be supplied as parameters to the system, therefore making it flexible and easily customised (Lee, 2002). An example of an in-house built trading system is the Stock Exchange Automated Trading System (SEATS) in Australian Stock Exchange (SEATS, 2003)).

2.2 Future of Trading Systems in Financial Markets

The internet has already changed the way many investors trade financial products by bringing up-to-date market information from Web and do-it-yourself trading online. For example, international brokers want to trade in multiple markets simultaneously wherever they are; diversified investors want to trade bundled portfolios. Trading systems in the future will be required to offer investors unprecedented convenience, choices and security in accessing fast updating information in order to make decisions in real time.

Traditionally and structurally, a trading system is a core part in financial markets. Therefore it plays a leading role in Business-to-Business (B2B) and e-business/e-Commerce-to-Enterprise (e2B) integrations. The development of an exchange service to tie other services in financial markets together becomes a compelling and urgent task for many trading firms and whole enterprises.

2.3 Exchange Service Requirements

Many new technologies will play fundamental roles to achieve those intentions and integrations in the information-intensive financial market industry. For an exchange service, there will be two leading aspects in quality of service: performance and security.

2.3.1 Performance Requirements

Trading engines are performance-critical systems and performance has always been a main key to their success. The transaction rates vary largely during different periods of day or year, when new securities are introduced. In some rare cases, it could go up to more than 60,000 orders/sec. Given the possibly large volumes of data in a short time and the need for timely dissemination, integration between a real-time trading system and other pre-trade and post-trade processes is the most important requirement in a financial market domain. In this case, the connection of a trading system with other applications should maintain an acceptable level of efficiency, more precisely saying that around 10% of performance degradation should be sufficient.

2.3.2 Security Requirements

Security is extremely critical in the finance industry trading systems. Financial market information infrastructure should especially provide the following basic security features:

- **Authentication and authorization.** If an exchange service receives an order from a trader, it must be able to identify the trader and his/her privileges. Likewise when the trader receives any information from the exchange service e.g. trade confirmation; the client side systems should be able to determine that the confirmation comes from the exchange service, as a trusted service, not from other sources.

- **Data integrity.** Buy or sell messages sent from investors to an exchange service travel through multiple routers over the open/private network. The system should make sure that orders are the same as the ones the exchange service receives.
- **Confidentiality.** Many traders require strong anonymity and high confidentiality. A confidential financial data leak can cause substantial damage to participants and ruin the trading market reputation. The most popular and effective approach to ensure confidentiality on a public network is through encryption.
- **Non-repudiation.** When brokers submit orders, they want to be assured that the exchange service does receive an identical order and can provide a proof to a third party to avoid disputes in case the order does not "go through" for any reason.
- **Denial-of-service (DoS) attack prevention.** A DoS attack could jam financial services or their communication channels with a huge amount of bogus data to prevent the system/service from responding to normal requests.

Other features like shill fraud prevention and front-running prevention are addressed by some more specific methodologies (Long et al., 2003). Special security concerns may be raised on integrated financial markets. For example, multiple vendors who all have individual user authentication and authorization should be controlled by single sign-on schemes.

3 EXCHANGE SERVICE DESIGN AND IMPLEMENTATION

We choose to design and implement our exchange service according to the principles of Web Service Architectures (Web Service, 2003). Technically described, such architectures are based on loosely coupled, self-defining and service-based software components communicating in XML-based messages over Internet standard protocols. More specific definitions can be found at the World Wide Web Consortium (W3C). Web services enable enterprises to expose their internal applications on the Web and make them accessible to business partners and broader communities. Its introduction in financial markets could facilitate inter-enterprise business processes, such as automated straight-through processing (STP).

As mentioned in the introduction, the proposed Exchange Service is based on a commercial trading system call X-Stream. We briefly overview the X-Stream system, before describing the design and implementation of the Exchange Service,

3.1 Architecture of X-Stream

X-Stream is one of first generation exchange trading systems that have been designed around a client-server architecture (Trading Technology, 2003). It has been used extensively as critical business application for many years. It is a proprietary system with a number of distinctive components such as:

- **Trading engine:** This is the essence of X-Steam. It performs order management, trade matching, reporting and circulation of market and trading data. Relational databases (Informix) are used to keep all necessary pre-trade and post-trade related information. A rich set of Application Program Interface (API) functions are provided for client and server application development.
- **Backup trading engine:** It serves as a standby trading engine in the event of trading engine failure, where it can be used as a primary trading engine transparently.
- **Distribution gateways:** They provide scalability, fault tolerance and market data distribution to clients.
- **Trader workplace:** It provides a client environment for brokers and users accessing exchange market sensitive information and submitting orders.

Built with Object-Oriented and relational database design methodologies (Jessup, 2002), X-Stream is highly configurable through Informix database parameters and is re-configurable in real-time via market control transaction. It is a huge, comprehensive, closely-coupled and mature system written in C++.

3.2 Overview of the Exchange Service

Our goal is to enable the integration of the trading system, in a seamless way, with some current and future applications, e.g. Web applications. We also plan to allow the system to be used in trading over the Internet, which becomes more and more popular among investors and brokers.

The Exchange Service makes X-Stream more visible and accessible for interaction with other applications, and is not meant to be a replacement of X-Stream.

The main advantage of using Web service technology is that it does not require any modification to the legacy systems. However its integration is not a trivial task. It depends on internal applications and business processes from various back-end systems.

A very feasible method for integrating X-Stream is based on an access wrapper. Figure 1 depicts the Exchange Service stack reference architecture.

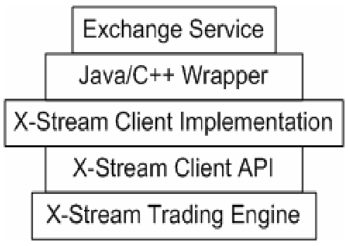

Figure 1: The layered architecture

Since X-Stream and its Client API are designed in Object Oriented style, the Client Implementation layer is drafted in the same fashion. The Exchange Service can be invoked in Client-Server fashion. For example a client process can place an order by requesting the appropriate service from the Exchange Service. More details are highlighted in the next subsections.

3.3 Wrapper/Client Implementation

Because Java has facilitated several popular, easy-use e-commerce enabling technologies, we first considered developing a Java wrapper allowing Java methods to invoke C++ codes by using the Java Native Interface (JNI), a part of the Java 2 Platform, Standard Edition (J2SE) (J2SE, 2003)

However, the overheads introduced by the Java wrapper made the frequently used functions almost five times slower than the original functions from X-Stream performance. This is unsatisfactory for a real time exchange service. We found that JNI related operation is the main reason for unsatisfactory performance. So we believe that we have to adapt a C++ wrapper instead of Java wrapper for the exchange service.

Web services are popular partially because of the XML-based SOAP protocol, which brings a powerful and versatile message exchange format. So we choose a SOAP development environment, gSOAP, to build a SOAP/XML web service.

gSOAP, an open source (gSOAP, 2003), provides a platform-independent development environment for deploying efficient SOAP/XML Web services in C and C++. Also there are other reasons for selecting it as a development tool (Engelen, 2003):

- Rapid application development (RAD) by automatic mapping XML-C++ and unique SOAP-C/C++ binding:
- Efficiency, eg precompiled RPC stub and skeleton routines quicken runtime encoding and decoding; steaming technology.
- Low memory overheads, low SOAP RPC latencies, easy control of memory allocation and development for real-time system.
- Support for Secure Socket Layer (SSL).
- Ease of use, eg gSOAP SDK generates a WSDL specification; client applications become an extensively automated development.

The pre-compiled marshalling routines for native C++ and user-defined data types enable the integration of C++ legacy applications, in this case X-Stream within SOAP services and clients. Functions such as placeOrder, getQuote are exposed as SOAP remote service methods.

The performance test results will be discussed in section 4 and exhibit sufficient outcomes which are compatible with performance of another component, the surveillance service (Dabous et al., 2003), in our financial market architecture.

3.4 Security

Although Web Services raise many security challenges, they can also function as powerful and flexible security tools. We investigated the possibilities of providing security as a shared service among our other services (Kong, 2002) and offer security service as utilities. This could separate the business functions of information service providers from security service providers – and let users choose and pay for the security service they need (Long et al., 2003). The preliminary results show significant benefits for cost control, maintainability, interoperability, visibility and manageability. However there are also disadvantages such as multiple standards, support technologies and possible bottlenecks with non-scalable solutions etc. So we decided to concentrate on authentication and data integrity by using digital signature and secure communication in SSL. Because of limited space, we only sketch our experience with SSL briefly.

Web Services have no built-in security model. More specifically, SOAP does not define vocabulary elements to transport security references from requesters to service providers and Web Services Description Language (WSDL) does not define elements to describe security capabilities and requirements. Therefore we must add a security protocol such as SSL (Freier, 1996) to meet the security requirements. In general, we believe that Public Key Infrastructure (PKI), for supporting digital signatures and document encryption and HTTPS/SSL, for secure point-to-point communication with known trusted parties will provide basic security for our application systems.

We chose SSL because it is:

- The main protocol for Web security;
- Fairly mature with almost a decade of improvements;
- Very widely implemented in many open source development kits; and
- Easy to use and to be deployed.

SSL is the current standard method of securing Web transactions. However it involves essential mathematical computations that take up CPU cycles and becomes a major cause of performance degradation. Some performance results will be discussed in section 4.

3.5 Other Work

Besides the development of a secure Exchange service, we have been working on three other directions.

Firstly, X-Stream has no push communication model, which means that brokers are not informed automatically when their orders become trades in the exchange system. Therefore we build a Peer-to-Peer Web Service on top of the Exchange Service for trade data disseminations. This will extend the functionality of X-Stream by informing brokers automatically.

Secondly, as gSOAP only supports request-response messaging, we are investigating the use of the Financial Information Exchange (FIX) protocol whose session is defined as a “bi-directional stream of ordered messages between two parties” (FIX Protocol, 2001). This means that there are no request-response semantics imposed by the specification. The Exchange Service could choose to apply a combination of one-way and notification interaction patterns rather than use Remote Procedure Call (RPC)-style communication in terms of implementing trading data dissemination scenario.

Thirdly, the Universal Description, Discovery and Integration (UDDI) is basically a service registry between the web service requestors and the web service providers (UDDI, 2002). In our case, the UDDI behaves as a locator of our Exchange Service for clients to discover its IP, port numbers WSDL-described services (WSDL, 2003).

Additional details on the Exchange Service design and implementation can be found in (Yip and Mok, 2003).

4 EXCHANGE SERVICE EVALUATION

This section discusses performance and security aspects of the proposed Exchange Service.

The first set of experiments addresses the concerns of performance of Web services (Vaughan-Nichols, 2002). The main reason is that XML is text-based rather than binary-based like CORBA's Internet Inter-ORB Protocol (IIOP), requiring more data to process (Davis and Parashar, 2002). The speed of coding and decoding and the message size are negatively impacted (Chiu et al. 2002).

We set up a performance testing environment within a Local Area Network (LAN). All exchange engines and gateways are supported by Dual Intel Xeon 2.8GHz processors. We adopt the classical methodology to test one of simplest functionalities: placing orders like brokers do in real life. We choose to submit one single order at a time only for the computing processes[1] in a few scenarios: one single client and multiple clients (up to 16) and four gateways to support load balancing.

The initial performance results (orders/sec) of Exchange Service are shown in Table 1. The performance overhead is generally defined as a ratio of absolute performance difference over the combined unit performance in percent. Take the example of 1 client case in Table 1, its overhead is (42.5-40.0)(orders/sec)/40.0(orders/sec) = 6.25%.

[1] In reality, most trading engines process orders in batch mode i.e. a group of order is processed at once.

Table 1: Preliminary performance results (orders/sec)

	Engine **without** Service	Engine **with** Service	Service Overhead
1 client	42.5	40.0	6.25%
4 clients	153.11	146.02	4.86%
8 clients	166.7	149.5	11.51%
12 clients	180.0	164.11	9.68%
16 clients	190.0	180.6	5.20%

Because the additional Web Service implementations increase memory usage during SOAP message processing, overheads must be due to SOAP messages that include numerous XML elements. The next experiment attempts to improve the performance and scalability of Web services using chunked transfers, HTTP keep-alive support, and increasing buffer size at both client and server sides. However the improvements on performance are not very significant. The results are depicted in Table 2.

Table 2: Enhanced performance results (orders/sec)

	Engine **without** Service	Engine **with** Service	Service Overhead
1 client	42.5	40.1	5.99%
4 clients	153.11	147.0	4.16%
8 clients	166.7	151.0	10.40%
12 clients	180.0	169.5	6.19%
16 clients	190	182.6	4.05%

Overall, the results show that the Exchange Service has not much negative impact on the performance of X-Stream.

Since security is another imperative but non-functional requirement for trading systems, the next experiment benchmarks the performance of a fully SSL-protected Exchange Service.

The added security protocol SSL slows the performance considerably. This is due to substantial processing time taken on encryption and decryption.

Table 3: with SSL performance results (orders/sec)

	Service **without** SSL	Service **with** SSL	SSL Overhead
1 client	42.5	5.6	658.93%
4 clients	153.11	6.5	2255.54%
8 clients	166.7	5.7	2824.56%
12 clients	180.0	5.85	2976.92%
16 clients	190.0	6.95	2633.81%

Table 3 shows of two orders of magnitude performance decrease, which is consistent with experiments conducted by another commercialising company, e.g. Preact Ltd. (Preact, 2003). Some improvements are recommended by Rescorla in (Rescorla, 2001).

Obviously the results, e.g. around six times slower in 1 client case, shown in Table 3 are unacceptable in reality. Currently we're considering element-wise encryption (with SOAP messaging), eg only sensitive parts should remain encrypted from beginning to end or in a more general manner of providing different levels of protection according to specific requirements. It will considerably increase the performance by reducing the noncompulsory computation overheads.

The choice of a Web server with better tuning of SSL transactions and cryptographic library on algorithm, suite on balance between security and performance all are significantly impact the speed (HPSSLperf, 2002).

A detailed experiment description and analysis can be found in (Yip and Mok, 2003).

5 CONCLUSION AND FUTURE WORK

From our development experiences, we ascertain that Web Services are suitable for dynamic integration with high availability, open standard and rapid engineering approaches.

The current Exchange Service intimately meets the requirement of a trading system in performance. A similar conclusion is drawn by (Kohlhoff and Steele) in terms of SOAP efficiency only. We are planning to continue our experiments with realistic data to compare the results between X-Stream and the Exchange Service. More work is required to address and ensure that both privacy and security can be preserved while customers, brokers, traders and researchers access the financial information easily, quickly and confidently.

In current financial market, a transaction can bind concurrent related connections to multiple parties and several layers of agents. For example, the trading engine could simultaneously connect with other brokers for placing orders, banks for settlement or credit checking, financial institutes for registry or market analysis etc. However SSL does not support multiple-party communications very well and cannot provide assurance of nonrepudiation without a third-party server (Security Roadmap, 2002). We are researching and experimenting secure XML protocols, e.g. XML Digital Signature (DSIG). The XML Signature standard provides a means for signing parts of XML documents. (XML Signature, 2003) and XML Encryption (XML Encryption, 2003) supported by W3C for efficient and secure data communication to complement SSL.

The XML Encryption standard permits encryption of portions of the message allowing header and other information to be clear text while leaving the sensitive contents encrypted to the ultimate destination with true end-to-end confidentiality. In web services environment, a service provider may play the role as a service requestor who sends information to multiple services.

Another group lead by Pradeep Ray (Ray, 2003) is collaborating with us on DoS attack prevention. The DoS attacks, in distributed forms have to be dealt with at the network management level.

Besides the security focus, in order to leverage our existing web services and to achieve an efficient, dynamic integration solution, a new approach of user-centric, process-driven and real time oriented service is designed and implemented in our lab. It is called real time trade data service (Cheung and Wu, 2003).

Another component in our architecture is a composite broker service for trading larger orders. This includes searching existing databases, generating trading plans based on output of analytics services, placing the orders and receiving executing results from trading service (Dabous and Lee, 2003).

There are some integration issues that must also be addressed in order to tightly coordinate with other organizations, individuals and working groups for customer satisfaction, operational excellence and legal requirements (Rabhi and Benatallah, 2002). For instance, settlement, which happens after trade completion, involves transferring funds between buyers' and sellers' bank accounts, needs to be promptly facilitated among different financial institutes. Currently, STP occurs a few days later after trading. The industry goal is T+0 (same day settlement). We are targeting various information technical solution options to explore and provide the STP opportunity for Australian financial market service sectors.

ACKNOWLEDGEMENT

We would like to sincerely acknowledge the contribution of many people on this project. We thank Stanley Yip, Joshua Mok, Chris Burgess, Johan Fischer, Yun Ki Lee, Glen Tan, Robert Chu Sunny Wu, Anthony Cheung, Rafal Konlanski and Jamaria Kong, Anne-Laure Mazon. The work is partially supported by Capital Markets Cooperative Research Centre (CMCRC, 2003).

REFERENCES

Allen, H. et al., November 2001. *Electronic trading and its implications for financial systems.*
At http://www.bis.org/publ/bispap07.htm

Capital Markets Cooperative Research Centre(CMCRC), 2002-2003. At http://www.cmcrc.com/

CBDI, 2003. At http://www.cbdiforum.com/index.php3

Cheung, A. and Wu, S., 2003. *A Trade Data Service,* Thesis, the University of New South Wales.

Chiu, K. et al., 2002. *SOAP for high performance computing,* Technical report, Indiana University. At http://www.extreme.indiana.edu/xgws/papers/soapPerfPaper/soapPerfPaper.pdf

Computershare, 2003. At http://www.computershare.com

Dabous, F. et al. 2003. *Performance Issues in Integrating a Capital Market Surveillance System using Web Services.* In proc. of 4th International Conference on Web Information Systems Engineering, Roma, Italy, Dec 2003

Dabous, F and Lee, Y., 2003. *Web Services Composition in Capital Market Systems*, Technical report, the University of New South Wales.

Davis, D. and Parashar, M., 2002. *Latency performance of SOAP implementations,* in proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid

Engelen, R. et al., 2003. *Developing Web Services for C and C++*, IEEE Internet Computing, March|April 2003 pp53-61

Exchange Service, 2003.
At http://129.94.244.61:8080/fit/prototypes.html/

FIX Protocol, 2001. The Financial Information Exchange Protocol (FIX), version 4.3, At
http://www.fixprotocol.org/specification/fix-43-pdf.zip

Freier, A. et al., 1996. *The SSL Protocol, Version 3.0*. Internet Draft.

At http://wp.netscape.com/eng/ssl3/draft302.txt

gSOAP, 2003.

At http://www.cs.fsu.edu/~engelen/soap.html

Harris, L., 2002. *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press

Hendershott, R., 2003. Electronic Trading in Financial Markets. In *IT Pro, July|August 2003.* ICCC Computer Society.

HPSSLperf, 2002. *Delivering the world's fastest HP-UX 11i SSL performance with the Intel®Itanium®2 processor family,* At http://www.hp.com/

Jessup, P., 2002. *Product Overview for Computershare X-Stream*, Computershare Technology Services Pty Ltd.

Jessup, P., 2002. *User Manual for Computershare ASTS*, Computershare Technology Services Pty Ltd.

J2SE, 2003*. Java 2 Platform, Standard Edition (J2SE),*

At http://java.sun.com/j2se/

Kohlhoff, C. and Steele, R., 2003, *Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems,* In proc. of The Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, May, 2003

Kong, J., 2002. *Security in Inter-domain Financial Market System Integration,* Thesis, the University of New South Wales

Lee, Y., 2002. *Design of Capital Market Systems*, Thesis, the University of New South Wales

Long, J. et al., July|August 2003. *Securing a New Era of Financial Services*, IT Pro

Preact Ltd., 2003. *SSL Performance,*

At http://www.preactholdings.com/performance/products/jetnexus/jet-nexus/SSL/

Rabhi, F. and Benatallah, B., 2002. *An Integrated Service Architecture for Managing Capital Market Systems*, IEEE network, 16(1)pp15-19

Ray, P., 2003, *Integrated Management from E-Business Perspective*, International Kluwer Academic/ Plenum Publishers

Rescorla, E., 2001. *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley.

SEATS, 2003.

At http://www.asx.com.au/markets/l4/seats_am4.shtm

Security Roadmap, 2002. *Security in a Web Services World: A Proposed Architecture and Roadmap – A joint security whitepaper from IBM Corporation and Microsoft Corporation.* At

http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/

Trading Technology, 2002. *Trading Technology Survey of Exchange Technology 2002*. At

http://www.tradingtechnology.com

[Accessed in August 2003]

UDDI, 2002. UDDI Version 3 Specification,

At http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3

Vaughan-Nicols, S., 2002. *Web services: Beyond the Hype*, IEEE Computer, 35(2) pp18-21

Viney, C., 2000. *McGrath's Financial Institutions, Instruments and Markets*. McGraw Hill. Sydney. 3rd edition

Web Service, 2003. *Web Services Architecture*, At http://www.w3.org/TR/2003/WD-ws-arch-20030808/#whatis

WSDL, 2003. *Web Services Description Language (WSDL) Version 1*.2, At http://www.w3.org/TR/wsdl2/

XML Encryption, 2003. *XML Encryption WG*,

At http://www.w3.org/Encryption/2001/

XML Signature, 2003. *XML Signature WG*,

At http://www.w3.org/Signature/

Yip, S. and Mok, J., 2003. *An Exchange Web Service*, Thesis, the University of New South Wales.