Standards for Securities Transactions

A Comparison Between Different Protocols

Stefan Nyström

TRITA-NA-Eyyyn

Master’s Thesis in Computer Science (20 credits)
at the School of Computer Science and Engineering,
Royal Institute of Technology, October 2002
Supervisor at Nada was Viggo Kann
Examiner was Stefan Arnborg
Abstract

In the financial sector, many different protocols are used for exchange of information related to securities transactions. At the time of writing (September 2002), the FIX protocol and the ISO 15022 protocol were the ones dominating the market. These protocols were focused on different stages of the trading life cycle making it hard to efficiently handle a complete trade. However, a new initiative had been taken by many actors in the financial sector to support the development of a new protocol called ISO 15022 XML. The new protocol was aimed at covering all areas covered by existing protocols and at eliminating the gaps between different areas of trading.

As a case study, evaluating the FIX and the ISO 15022 XML protocols, I designed and implemented a translator between these two protocols. The work on the translator revealed many interesting aspects of the protocols. These aspects included how the messages were structured in the two protocols and what tools were available when translating messages of these protocols. The case study also brought some of the obstacles in migrating from FIX to ISO 15022 XML into focus. The problem areas in this process were mainly in the different areas covered by the two protocols.

Standarder för värdepappershandel

En jämförelse mellan olika protokoll

Sammanfattning


Contents

1 Introduction 1

2 Previous Work in the Area 3
  2.1 Description of Various Protocols . . . . . . . . . . . . . . . . . . . 3
     2.1.1 The FIX Protocol . . . . . . . . . . . . . . . . . . . . . . . . 3
     2.1.2 ISO 15022 and the SWIFT Message Standard . . . . . . . . 5
     2.1.3 The ISO 15022 XML Protocol . . . . . . . . . . . . . . . . . 5
     2.1.4 The ORC System . . . . . . . . . . . . . . . . . . . . . . . . . 7
     2.1.5 The eXtensible Markup Language (XML) . . . . . . . . . . . 7
  2.2 Description of FIX-Link . . . . . . . . . . . . . . . . . . . . . . . . 8

3 Questions About the FIX and ISO 15022 XML Protocols 9

4 Working Methods Used in the Project 10
  4.1 Literature Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  4.2 Case Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

5 Choice of Implementation Approach 12
  5.1 Restricting Issues . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  5.2 Writing a Replacement of Fenix . . . . . . . . . . . . . . . . . . . . . 12
  5.3 Writing an ISO 15022 XML to FIX 4.2 Translator . . . . . . . . . . 13
  5.4 Translating ISO 15022 XML Messages Using Fenix’ API . . . . . . . 14
  5.5 Final Decision . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

6 How the System Was Implemented 16
  6.1 The FIX Engine . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  6.2 The ISO 15022 XML Engine . . . . . . . . . . . . . . . . . . . . . . 17
  6.3 The Message Translators . . . . . . . . . . . . . . . . . . . . . . . . . 18

7 Results 22

8 Conclusions 25
  8.1 Advantages of ISO 15022 XML . . . . . . . . . . . . . . . . . . . . . . 25
  8.2 Cost of a Transition to ISO 15022 XML . . . . . . . . . . . . . . . . 25
  8.3 Problems Faced in the Development Process . . . . . . . . . . . . . 26
  8.4 Recommendations for Future Work . . . . . . . . . . . . . . . . . . . 27

References 28

A Class Diagram of the Developed Application 30
List of Figures

2.1 An example of a class diagram in UML. The diagram shows how an IndicationOfInterest class can contain SpreadToBenchmark and SwapSpread classes. .................................................. 6
2.2 A picture of how FIX-Link connects to a client and an ORC system. 8
5.1 A visualization of how a replacement of Fenix would work together with the remaining system. ...................................................... 13
5.2 A picture of how an ISO 15022 XML to FIX 4.2 translator could work together with FIX-Link. ................................. 14
5.3 Storing ISO 15022 XML messages using Fenix’ API. The new translator would translate ISO 15022 XML messages directly into Fenix’ API. ............................... 15
6.1 Detailed layout of the ISO 15022 XML to FIX 4.2 translator. The translator was divided into three parts: the ISO 15022 XML engine, the FIX engine and the message translator that lay between them. The message translator was itself divided into two parts: iso2fix which translated ISO 15022 XML messages into FIX messages and fix2iso which did the opposite. ........................................ 17
6.2 Visualizes how Fenix can be used as FIX engine for the new translator. 17
6.3 Class diagram of the ISO 15022 XML API. IsoXmlMessage classes could contain IsoXmlMessageComponent and IsoXmlMessageElement classes. IsoXmlMessageComponent classes could contain other IsoXmlMessageComponent classes or IsoXmlMessageElement classes. IsoXmlMessageElement classes contained only simple data. .............................. 18
6.4 Class diagram of how interested classes are notified using the observer design pattern. .......................................................... 19
6.5 Class diagram of how the FIX to ISO 15022 XML translators collaborated. The FixTranslatorFactory contained one FixTranslator for each message type supported. ........................................... 20
7.1 Differences in business areas and parts of the trading process supported by various protocols. The aim of the ISO 15022 XML protocol was to support all areas covered by existing protocols. .................... 24
## List of Tables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of a FIX message of type Indication of Interest. A trading party is interested in selling shares in Ericsson.</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparison between FIX and ISO 15022 XML. The two messages have the same business content.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Before modern technical devices like telephones and computers were in general use, stock trading took place on a physical market place. On the market place brokers could meet and deal in stock. The customers had to come to the market place to meet with the brokers and discuss what they were interested in. When phones became readily available, customers could use the phone as a means of placing orders with their broker. Today, the market is often controlled by a computer system that handles orders and ensures that the customers get the best price for their stock. Big customers trading a lot of stock can streamline their process and avoid making phone calls to their stock broker by using protocols like the FIX [4] and ISO 15022 protocols [7].

This thesis deals with a master’s project for the School of Computer Science and Engineering at the Royal Institute of Technology (KTH). The project was executed at the Department of Numerical Analysis and Computer Science (NADA) for Computer & Audio-Technical Systems CATS AB.

The master’s project was concerned with standards for communication of information related to securities transactions. It focused on the ISO 15022 XML protocol (a new protocol being developed) and a transition from the FIX protocol to ISO 15022 XML. The main part of the project was a case study where an existing FIX enabled application was adapted to support ISO 15022 XML.

Chapter 2 describes the different protocols used today and introduces ISO 15022 XML. FIX-Link, a product that played a major part in the master’s project, is also presented in this chapter. Most of the information presented here comes from the literature study performed at the beginning of the master’s project. Chapter 3 discusses the focus of the master’s project and which questions the project was aimed to answer. Later chapters introduce the case study and how the existing FIX application was adapted to support ISO 15022 XML. They describe the decisions made and what problems were faced during development. In the last chapters, the results of the case study, an analysis of the results and some recommendations for further work are presented.

Throughout this thesis I make frequent use of the words pre-trade, trade, post-trade and settlement. These words indicate stages in the trading process. Pre-trade
is the stage at which trading parties advertise their interest in e.g. buying or selling stock. Trade is the process in which buyer and seller agree on the price and amount of the trade. Post-trade and settlement are the stages where the deal is closed and the actual transaction takes place.
Chapter 2

Previous Work in the Area

This chapter gives some background information and discusses some of the protocols used at the time of writing for exchange of trading information. It also introduces the ISO 15022 XML protocol which is the protocol on which the thesis centres. At the end of the chapter is a short description of FIX-Link which is a product developed at CATS AB.

2.1 Description of Various Protocols

2.1.1 The FIX Protocol

The Financial Interface eXchange (FIX) protocol [4] [5] is an open message standard for exchange of information related to securities transactions. The work on the FIX protocol was initiated in 1992 by a group of American institutions and brokers who together formed FIX Protocol Limited (FPL). The aim was to streamline their business by automating information exchange between them and their customers. The customers would no longer have to make unnecessary phone calls to place orders [4]. The FIX protocol has grown strong especially in the pre-trade and trade areas, i.e. messages that are communicated prior to an order and messages that are used to place orders and communicate the status of orders [12].

The protocol consists of two different types of messages: the administrative messages and the application messages. The administrative messages, or session level messages, are concerned with the secure delivery of data and include logon, logout and heartbeat messages\(^1\). These messages allow FIX to be carrier independent enabling a FIX session to run over e.g. a leased line or TCP/IP.

Every FIX message has a sequence number that identifies it. It is the responsibility of the receiving application to detect gaps in the sequence numbers [4]. When it does it sends a resend request message instructing the communication partner to re-send the missing messages.

\(^1\)Heartbeat messages are sent periodically to indicate that the connection is active.
Table 2.1. Example of a FIX message of type Indication of Interest. A trading party is interested in selling shares in Ericsson.

<table>
<thead>
<tr>
<th>Message part</th>
<th>Message</th>
</tr>
</thead>
</table>
| Message header | 8=FIX.4.2  
9=154  
35=6  
49=BRKR  
56=INVMGR  
34=236  
52=19980604-07:58:48 |
| Message body | 23=115685  
28=N  
55=LME  
54=2  
27=200000  
44=10100.000000 |
| Message trailer | 10=159 |

The application messages are the carriers of data in the FIX protocol and include the NewOrderSingle, ExecutionReport and OrderCancelReject messages. It is through these messages that a trading party can e.g. place orders and get information about the status of orders.

Every FIX message consists of a number of tag-value pairs where the tag is identified by a tag number. Below is an example of a tag-value pair in FIX.

49=BRKR

In this example, 49 is the tag number and BRKR is the value. Tag-value pairs are delimited using a special character.

All messages are divided into three parts: the message header, the message body and the message trailer. The message header holds e.g. the message sequence number and a tag that defines what type of message it is. The message body carries all the business information that is to be communicated while the message trailer only holds a checksum. The checksum is used to calculate if the message has been received intact.

Table 2.1 shows an example of a FIX message of type Indication of Interest which is a typical pre-trade message. The first tag-value pair, 8=FIX.4.2, identifies the beginning of a new message and the protocol version. The value of the second tag, 9=154, is the length of the message in bytes and the value of tag 35 identifies this as a message of type Indication of Interest [4]. The next two tags identify the firm sending the message and the firm receiving the message. Tag 34 is the message
sequence number while the last tag of the message header is the time when the message was sent.

The first tag in the body of the example message is a unique identifier for the Indication of interest. It is used for later referencing of the message. Tag 28 declares that this message is a new message and not a resend or cancel message. The next tag defines what shares the message refers to. In this case someone is interested in shares in Ericsson. Tag 54 claims that they are interested in selling shares. The number of shares to be sold is defined by tag 27 and the price by tag 44. The only tag in the message trailer is the checksum of the message.

2.1.2 ISO 15022 and the SWIFT Message Standard

The ISO 15022 standard was created by Technical Committee ISO/TC68 (Banking, Securities and Related Financial Services) Sub-Committee SC4 (Securities and Related Financial Instruments) [7]. The protocol replaces previous standards for electronic message exchange like ISO 7775, Scheme for message types, and 11521, Scheme for interdepository message types [7]. The reasons these protocols were replaced were mainly that they were considered to be too restrictive and ambiguous [1].

ISO 15022 is essentially a set of data types, defined in a data field dictionary, describing for example price, date and quantity. These data types are used to form messages that can be registered by The Society for Worldwide Interbank Financial Telecommunication S.C. (SWIFT) [14] which is the registration authority for ISO 15022 messages [7]. As registration authority, SWIFT is responsible for the data field dictionary and makes sure that new messages are consistent with the standard.

Apart from being the registration authority for ISO 15022, SWIFT has also defined a set of data types and messages based on ISO 15022 that are the de-facto standard for communication of information related to securities transactions in ISO 15022. The messages and data types defined by SWIFT are mostly used in the post-trade area of securities transactions, i.e. after the price has been agreed on and the actual transaction is to take place.

Information in ISO 15022 is represented using tags with specified format. Tags representing related information are grouped into sequences and sub sequences. A message usually contains several sequences making up a slightly hierarchical structure.

ISO 15022 messages are mainly exchanged on SWIFT’s private network SWIFT-Net. The network traffic runs over X.25 and all session handling is done by SWIFT internally [1].

2.1.3 The ISO 15022 XML Protocol

In July 2001 FIX Protocol Limited and SWIFT decided to seek convergence of their respective protocols through supporting the development of a new protocol called ISO 15022 XML [3]. The protocol is currently being developed by ISO Working
Group 10 [8]. It is based on ISO 15022 but is aimed to cover all areas currently covered by other existing protocols such as the FIX protocol.

SWIFT will work as the registration authority for ISO 15022 XML as well [9]. As such it will be responsible for adding new messages to the standard as well as for storing all messages and data types in a message repository.

The protocol adopts the use of the Unified Modelling Language (UML) [11] as a way of modelling and representing messages in the repository. UML is a graphical way of representing e.g. object oriented systems. While UML includes several different types of diagrams, the type used to represent ISO 15022 XML messages is called class diagrams. Class diagrams usually picture how different classes are built up of other classes and how classes collaborate. Figure 2.1 shows a typical class diagram. It illustrates how an IndicationOfInterest class can contain SpreadToBenchmark and SwapSpread classes.

To enable a smooth transition from existing standards and to enforce complete coverage of the business areas and parts of the trading life cycle, the messages of different existing protocols are reverse engineered into UML message models [9]. The reverse engineering process is mainly focused on FIX for the pre-trade and trade parts of the trading process while post-trade messages are derived from ISO 15022 [6].

The standard way to send ISO 15022 XML messages is to use XML as the physical representation for the messages. Refer to section 2.1.5 for a short presentation of XML. Strict conversion rules from the UML message models to XML messages have been developed by working group 10 [10].
ISO 15022 XML messages are constructed in a hierarchical way. Messages consist of message elements and message components. Message elements are simple data types. The following is an example of a message element defining an ISIN code:

\(<\text{ISIN}>\text{GB2200376500}</\text{ISIN}>\)

Message components on the other hand represent more complex information than message elements and can contain both message elements and other message components. Here is an example of a message component representing a financial instrument symbol:

\(<\text{FinancialInstrumentSymbol}>\text{<ISIN>GB2200376500</ISIN> }<\text{AlternateSymbol}>\text{<ID>2200376500</ID> }<\text{DomesticIDSource>GB</DomesticIDSource}>\text{</AlternateSymbol> }\text{</FinancialInstrumentSymbol>}\)

### 2.1.4 The ORC System

The ORC system is not actually a protocol but a system for automatic placing of orders on trading exchanges [13]. The system connects to one or more trading exchanges allowing client applications to trade simultaneously on different markets. The ORC system manages orders, positions and completed transactions.

### 2.1.5 The eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) [15] is a structured way of representing data. XML documents are built up of nested start and end tags like the example below.

\(<\text{FinancialInstrumentSymbol}>\text{<ISIN>GB2200376500</ISIN> }<\text{AlternateSymbol}>\text{<ID>2200376500</ID> }<\text{DomesticIDSource>GB</DomesticIDSource}>\text{</AlternateSymbol> }\text{</FinancialInstrumentSymbol>}\)

As can be seen from the example, every start tag \(<\text{name}>\) has a matching end tag \(<\text{name}/>\). XML is a free format. No tags or data formats are specified by the standard itself. Instead, Document Type Definitions (DTDs), or XML Schemas are used. DTDs and XML Schemas specify what tags and sub-tags are allowed in an XML document. They also specify the format of the tags. ISO 15022 XML for example adopts the use of XML Schemas to specify how the messages are built.

---

2 An exception to this rule exists when start and end tags are combined to \(<\text{name} />\)
2.2 Description of FIX-Link

Computer & Audio-Technical Systems CATS AB has developed a product called FIX-Link [2]. FIX-Link works as a message translator that translates certain FIX messages into corresponding commands on an ORC System (See figure 2.2). The product consists of three main components:

**Orelix** which is the interface to the ORC system,

**Fenix** which is the FIX engine of the system, and

**Idefix** which is a graphical tool for monitoring and controlling the system.

As FIX engine, Fenix handles the FIX session, sends and receives FIX messages, and includes an application level interface to create, handle and validate FIX messages. FIX-Link operates on a subset of the FIX standard including all administrative messages as well as some common application messages.
Chapter 3

Questions About the FIX and ISO 15022 XML Protocols

The master’s project touched on several questions about the FIX protocol. These questions were mainly about what the FIX protocol was, how it was structured and what business areas and stages of the trading process it was aimed at covering.

However, the main goal of the project was to learn about ISO 15022 XML and the differences between the two protocols. What, for example, made it interesting and necessary to develop ISO 15022 XML and what were the main differences between the two protocols? Would ISO 15022 XML cover any new parts of the trading life cycle? What were the main advantages and disadvantages of the new protocol?

It was also vital to understand how costly a transition from the FIX protocol to ISO 15022 XML would be. The cost was of course related to the work effort required to make the transition. Would it be possible to create an automatic translator between the two protocols or did they differ too much in structure of the messages and business content? What aspects of the new protocol made it interesting for a company to adopt it?

At the time of writing, ISO 15022 XML was still under construction. It was thus also interesting to know how long it would be before the protocol reached maturity and was ready for implementation.
Chapter 4

Working Methods Used in the Project

The master’s project was focused on learning about the FIX protocol, the ISO 15022 XML protocol and the differences between them. The work effort in the project was divided between a literature study and a case study.

4.1 Literature Study

By studying the FIX standard documents [4] [5] I was able to learn much about how the FIX protocol was structured and what business areas and parts of the trading life cycle were handled by it. Likewise by reading the information released on ISO 15022 XML, many questions about this protocol could be answered. Interesting aspects of this protocol included why it was developed, what business areas and parts of the trading life cycle were covered by it, and how the messages were structured. Much of the information acquired in the literature study was presented in chapter 2.

4.2 Case Study

To learn about the differences between the protocols and evaluate how costly and hard it would be for a company to make the transition from the FIX protocol to ISO 15022 XML, it was decided that a case study was to be performed. The case study included writing an implementation of ISO 15022 XML in FIX-Link. As described above, FIX-Link was a product that translated FIX messages into appropriate commands on an ORC system. The actual case study included replacing the FIX engine of FIX-Link, Fenix, with an ISO 15022 XML enabled substitute. The resulting application would thus translate ISO 15022 XML messages into ORC commands. Through this process some of the differences between the protocols would become evident and much knowledge about the structure of the two protocols would be generated. It would also be a good example of how big changes a product would have to go through in order to make the transition from FIX to ISO 15022 XML. To make further value of the case study, all code would be developed keeping
in mind that it should be easy to expand and reuse in other projects concerning ISO 15022 XML.
Chapter 5

Choice of Implementation Approach

When developing an ISO 15022 XML enabled version of FIX-Link there were several aspects to consider. It was for example desirable to change as little of the original product as possible to minimise the effort in development. It was also vital that the work could be easily adapted to function as a clearly defined master’s project. Some more restricting issues on the implementation strategies are presented in section 5.1. Considering these things the three strategies presented in this chapter seemed most compelling.

5.1 Restricting Issues

At the time of writing, the ISO 15022 XML protocol was in its early stages of implementation. Thus, no information of the session handling facilities of the protocol had been released. The only available part of the protocol was a draft of the Indication Of Interest (IOI) message. An equivalent to this message was present in the FIX protocol but it was not currently supported by FIX-Link. There was no reason for FIX-Link to support this message since there was no equivalent of that message in the ORC system. The ORC system was not expected to change in this aspect and this added some restrictions on the possible implementation strategies.

5.2 Writing a Replacement of Fenix

The most natural approach in creating an ISO 15022 XML enabled version of FIX-Link was to substitute the FIX engine of FIX-Link, Fenix, with a new component designed to handle ISO 15022 XML messages (see figure 5.1). Since the application level interface (API) of Fenix was designed around the structure of a FIX message it was not suitable to represent ISO 15022 XML messages. The new component would thus have to have its own API. Orelix, the part of FIX-Link which was most tightly bound to Fenix would have to be largely rewritten in order to support this new API. On the other hand, the replacement for Fenix would probably be fairly simple
to write since there were several tools for managing and parsing XML documents available.

A problem with this approach was that FIX-Link did not support the Indication Of Interest message which was the only message of ISO 15022 XML available. Thus, the only useful parts of the approach that could be written in the scope of this thesis were the API for the ISO 15022 XML component and the classes used for parsing XML messages. No actual translation between protocols could be implemented before some more ISO 15022 XML messages were released.

The positive sides of this approach could be summarised as:

- The replacement of Fenix would be fairly simple to write.
- It was the most natural and efficient way to design a translator.

The drawbacks of the approach were summarised:

- Large parts of the existing system would have to be rewritten.
- No actual translation could be implemented at this stage.
- The master’s project would focus more on processing and parsing XML documents than on standards for securities transactions.

5.3 Writing an ISO 15022 XML to FIX 4.2 Translator

Another approach was writing a translator between ISO 15022 XML and FIX 4.2. The translator could work together with FIX-Link to translate ISO 15022 XML messages via FIX messages to ORC commands (see figure 5.2). The obvious advantage of this strategy was that existing code would not have to be altered. Another advantage was that the translator could work independent of FIX-Link as a product of its own.

A drawback to this approach was that when operating together with FIX-Link, some excessive work would be done in creating and parsing FIX-messages. The application would thus be less efficient in translating ISO 15022 XML messages into ORC commands. This solution could also be considered to be less elegant than the previous one since it involves translations to and from an intermediate protocol (FIX).
The approach had the advantage of being easily implemented as a clearly defined master’s project. The project would involve designing a foundation for message translators between different ISO 15022 XML and FIX 4.2 messages. Since the Indication Of Interest message existed in both FIX and ISO 15022 XML a translator of this message could be implemented as a model of how other message translators could be implemented.

The main benefits of the approach were:

- The existing code would not need to be edited.
- The system could work independently as its own product.
- It would constitute a suitable project for master’s thesis.

The disadvantages were:

- Unnecessary translations would be required when used with FIX-Link.
- It could be considered a less elegant solution.

5.4 Translating ISO 15022 XML Messages Using Fenix’ API

A third approach that was somewhat similar to the previous one was to parse the ISO 15022 XML messages directly into the Fenix API (see figure 5.3). Using this approach Orelix could be used without alterations, though Fenix would have to be changed to remove e.g. session control. The system would also have to be altered to change the way the different threads of execution worked together.

The main advantage of this method as opposed to writing a translator between ISO 15022 XML and FIX was that less overhead in the form of unnecessary session handling would be required. The main drawback on the other hand was that, like the first approach, not much translation could be implemented before more ISO 15022 XML messages were released.

5.5 Final Decision

After careful consideration of all approaches I decided to implement a translator between ISO 15022 XML and FIX 4.2. The basis for this decision was mainly that
this approach would be easiest to implemented as a clearly defined thesis project but also that the translator could be implemented as a product of its own. Much of the code written for this method could easily be reused if, at a later stage, one of the other approaches should be implemented.
Chapter 6

How the System Was Implemented

This chapter describes how the translator was implemented. It contains technical
details on what methods were used and how the system was designed. A reader not
interested in implementation details could skip to chapter 7.

It was decided that the programming language Java would be used for the
implementation of the translator between ISO 15022 XML and FIX. This decision
was founded on the fact that Java was object oriented and ran on many platforms.
Using Java also enabled me to easily reuse code from FIX-Link which was written
entirely in Java.

The ISO 15022 XML to FIX 4.2 translator was easily divided into three parts
(see figure 6.1). The first part, the FIX engine, handled all FIX specific issues
like parsing of FIX messages and controlling the FIX session. Since the application
should be easy to use in conjugation with FIX-Link it was vital that this part was
compatible with Fenix which was the FIX engine of FIX-Link.

Another part of the system, the ISO 15022 XML engine, handled the tasks which
were specific to ISO 15022 XML. These tasks included parsing and handling ISO
15022 XML messages. Since no information about the session level messages of ISO
15022 XML was released, no session handling in this part of the system could be
written in the scope of this master’s project. Making a clear distinction between the
different parts of the system would make it more easy to reuse the parts in other
projects. The ISO 15022 XML part of the system was thus implemented with a
clearly defined and documented API making it easier to use separately.

Between the FIX and the ISO 15022 XML specific parts of the system lay
the actual translator. The translator was divided into two parts: fix2iso which
translated FIX messages into ISO 15022 XML messages, and iso2fix which did the
reverse translation (see figure 6.1). Since the application should be constructed as
a base for further development and to make it easier to support new message types,
the message translators should be as easy as possible to develop.
6.1 The FIX Engine

One of the major parts that constituted the translator between FIX and ISO 15022 XML was the part that handled the FIX session and all other FIX specific tasks. Such a component is often called a FIX engine. There existed several different FIX engines on the market. An important aspect to consider when choosing one was that the engine should be compatible with Fenix, which was the FIX engine of FIX-Link. The choice fell naturally on using Fenix in this application as well, since an instance of Fenix would not require much configuration to connect to another instance of Fenix (see figure 6.2). Another aspect that influenced this choice was that Fenix was developed at CATS AB. The master’s project was also done at CATS and I could thus get much support on the use of Fenix throughout the project. Fenix handled all session level messages internally and offered an easy to use application level interface to receive, send and create FIX messages.

6.2 The ISO 15022 XML Engine

Another part of the system handled the parsing of ISO 15022 XML messages. This part was the ISO 15022 XML engine of the system. The engine consisted of three parts: a message parser, an API to handle messages internally, and a part for session handling and sending and receiving messages. Since no information about session level messages in ISO 15022 XML was available, no session handling could
be implemented in the ISO 15022 XML engine at this stage. The only parts of
the engine that could be implemented were the application level interface (API) to
create, send and receive ISO 15022 XML messages as well as the ISO 15022 XML
parser.

The API was centred on the IsoXmlMessage, IsoXmlMessageComponent
and IsoXmlMessageElement classes which made handling of ISO 15022 XML
messages easy. To reflect how the messages in ISO 15022 XML were built the
IsoXmlMessage class could contain IsoXmlMessageComponent and
IsoXmlMessageElement classes. IsoXmlMessageComponent classes
contained other IsoXmlMessageComponent classes and IsoXmlMessageElement
classes which in turn only contained simple data (see figure 6.3).

The parser for ISO 15022 XML messages was designed using the Simple API
for XML (SAX). SAX used an external XML parser that validated XML messages
using XML schemas. The SAX system was based on events notifying the applica-
tion when start and end tags in XML were read, as well as when simple data
was read. By responding to these events the parser class was able to build a
structure of IsoXmlMessage, IsoXmlMessageComponent and IsoXmlMessageElement
classes from the original message.

6.3 The Message Translators

Between the two protocol engines lay the actual translator. The translator was
divided into two parts. The fix2iso part translated incoming FIX messages into
ISO 15022 XML messages and the iso2fix part did the reverse. A picture of the idea
can be found in figure 6.1 on page 17. The two parts were implemented as two separate
threads running in parallel. They worked very similar and included a translator.
class for each incoming message type supported. The system was designed holding in mind that these translator classes should be as easy to implement as possible making it efficient to upgrade the product to support new message types.

When a new message arrived, interested classes were notified using the observer design pattern (see figure 6.4). There was only one observer present in the system, but the design pattern was chosen to allow expansion of the program to include for example more logging facilities.

For each incoming message, a MessageTranslateListener class received notification about the message and chose an appropriate message translator for the message by checking the message type. For example, if a FIX message of type Indication Of Interest was received, the MessageTranslateListener class invoked the IndicationOfInterestTranslator class to translate the message.

Since the message translator classes had little need for internal state and instantiation of classes was time consuming in Java, it was decided that there should be only one instance of each message translator class in the application. This single instance was invoked for every new message (of the appropriate type) received. A message translator factory handled the instances of the message translator classes (see figure 6.5).

The message translators were meant to be designed following a special pattern. Since there was only one message translator instance for each message type supported in the application the message translators were not allowed to have internal state. All message translators also had to implement the MessageTranslator interface in order to be compatible with the rest of the system. Additionally, message translators could subclass the MessageTranslatorBase class to get access to some common functionality of message translators (see figure 6.5).

The message translators translating FIX messages into ISO 15022 XML messages usually had one method for every message component that was to be created in the new message. There was nothing in the system that required this structure,
Figure 6.5. Class diagram of how the FIX to ISO 15022 XML translators collaborated. The FixTranslatorFactory contained one FixTranslator for each message type supported.
but dividing the translating process over many methods made it more foreseeable.
The code below illustrates this concept.

    ...  
    public void translateMessage(Message fixMessage)  
    ...  
    protected IsoXmlMessageComponent  
        createFinancialInstrument(Message fixMessage)  
    ...  
    protected IsoXmlMessageComponent  
        createFinancialInstrumentDetails(Message fixMessage)  
    ...  
    protected IsoXmlMessageComponent  
        createProductType(Message fixMessage)  
    ...  

    The message translators of the iso2fix part of the system on the other hand usually
    had one method for each message component of the incoming message that was to
    be translated. This structure was, like for the fix2iso translators, chosen to make
    the translation more foreseeable. The code below illustrates the concept.

    ...  
    public void translateMessage(IsoXmlMessage message)  
    ...  
    protected void  
        translateFinancialInstrument(IsoXmlMessageComponent fi,
                                           Message ioi)  
    ...  
    protected void  
        translateFinancialInstrumentDetails(IsoXmlMessageComponent fid,
                                           Message ioi)  
    ...  
    protected void  
        translateProductType(IsoXmlMessageComponent productType,
                                           Message ioi)  
    ...  
    ...
Chapter 7

Results

At the end of the master’s project, the translator between FIX and ISO 15022 XML worked satisfactory. I was able to set up an instance of Fenix to communicate indication of interest messages with the new application on my computer. The application was able to translate most fields of the respective protocols (see chapter 8 for an explanation on why all fields could not be translated).

Through the work on the translator some issues about the protocols became evident. There was for example a big difference in how the messages of the two protocols were structured. The messages of the ISO 15022 XML protocol were highly hierarchical and consisted of nested message components and message elements. The FIX protocol on the other hand had a very flat structure where all tags were basically on the same level. This difference is visualised in table 7.1. The table shows the message bodies of an indication of interest message for the two protocols. The two messages in this table express the same information and the ISO 15022 XML version was actually generated by feeding the FIX version through the translator developed in this project. From the table it is also evident that the ISO 15022 XML format of messages was longer and less space efficient than the FIX message format.

Another major difference between the protocols was in the areas of the trading process covered. ISO 15022 XML would, when it was finished, cover both the pre-trade, trade and post-trade areas while the FIX protocol covered only pre-trade and trade (see figure 7.1).

Message components in ISO 15022 XML were meant to be reusable between different messages. To support this reusability, some message components were more complex and general in definition than they would have been if they were part of only one message [10]. However, the generality in definition constituted a problem when creating the translator from ISO 15022 XML to FIX. The definition of the message components gave a user the freedom to create messages in ISO 15022 XML whose content could not be translated into standard FIX messages. The FinancialInstrumentDetails message component for example contained the message element Rating which did not have a counterpart in FIX.
Table 7.1. Comparison between FIX and ISO 15022 XML. The two messages have the same business content.

<table>
<thead>
<tr>
<th>FIX</th>
<th>ISO 15022 XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>23=115685</td>
<td>&lt;IndicationOfInterest&gt;</td>
</tr>
<tr>
<td>28=N</td>
<td>&lt;MessageReference&gt;115685&lt;/MessageReference&gt;</td>
</tr>
<tr>
<td>54=LME</td>
<td>&lt;FinancialInstrument&gt;</td>
</tr>
<tr>
<td>55=2</td>
<td>&lt;FinancialInstrumentSymbol&gt;</td>
</tr>
<tr>
<td>27=200000</td>
<td>&lt;TickerSymbol&gt;LME&lt;/TickerSymbol&gt;</td>
</tr>
<tr>
<td>44=10100.00000</td>
<td>&lt;/FinancialInstrumentSymbol&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/FinancialInstrument&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;IndicationOfInterestDetails&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Side&gt;SELL&lt;/Side&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Quantity&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Unit&gt;200000&lt;/Unit&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Quantity&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Price&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;TED&gt;FALSE&lt;/TED&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;TradePrice&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Amount&gt;10100.0&lt;/Amount&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/TradePrice&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/Price&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/IndicationOfInterestDetails&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/IndicationOfInterest&gt;</td>
</tr>
</tbody>
</table>

23
Figure 7.1. Differences in business areas and parts of the trading process supported by various protocols. The aim of the ISO 15022 XML protocol was to support all areas covered by existing protocols.
Chapter 8

Conclusions

8.1 Advantages of ISO 15022 XML

After working with FIX and ISO 15022 XML there were some conclusions that I could draw about advantages and disadvantages of the two protocols. The main reason ISO 15022 XML was developed was probably a desire to be able to use the same protocol for all stages of trading. ISO 15022 XML thus supported both the pre-trade, trade and the post-trade areas. The biggest award for making the transition to ISO 15022 XML would be for companies already supporting or wanting to support the whole trading life cycle. These companies included those who were currently using for example both FIX and ISO 15022 in their business. With ISO 15022 XML they could integrate the pre-trade, trade and post-trade areas more efficiently. However, a company whose business was more focused on pre-trade and trade, which were the areas covered by the FIX protocol, could also benefit from adopting ISO 15022 XML.

It was pretty likely that the industry would be converging to ISO 15022 XML in time, and by adopting the protocol at an early stage a company would have a head start in the competition. It was difficult to see when ISO 15022 XML would be ready and reach maturity. The pre-trade and trade parts of the protocol would probably be released at the end of 2002 but there was no indication on when session level messages were going to be released. It was likely that it would take up to a year before the protocol was released in total and some years after that before the big companies would start to use it.

8.2 Cost of a Transition to ISO 15022 XML

One of the more interesting questions the master’s project was aimed at answering was how costly it would be for a company to adopt the new standard. The cost would of course depend much on what protocols were currently used and how they were used. The translator developed in this master’s project did not require a massive work effort. However, the application would require much more work and
testing before it was ready for industrial use. See section 8.4 to see what parts of
the application could not be completed.

8.3 Problems Faced in the Development Process

The translator read the whole contents of a message before translation of the mes-
sage took place. Thus the difference in the hierarchical structure of the ISO 15022
XML messages versus the flat structure of FIX messages had little impact on the
complexity of the translator.

Since the ISO 15022 XML protocol was aimed at covering many more areas
of the trading process than the FIX protocol currently did it was natural that a
translator would not be able to support all ISO 15022 XML messages. However,
this limitation was not necessarily a problem for a company moving from FIX to
ISO 15022 XML since such a company would not initially have the need for the
unsupported messages.

A problem that was faced during the development of the translator in this
master’s project was the fact that message components in ISO 15022 XML were
more general in information content than could be expressed in single FIX messages.
The translator was thus unable to translate the whole ISO 15022 XML messages
into FIX.

Workarounds for this problem could be created on a connection to connection
basis. The excess information could for example be translated into user defined
FIX tags. This solution is outside of the FIX standard and the user defined tags
would have to be agreed on by both communicating parties. Such a solution would
probably not be considered a long term one. A user defined FIX tag would typically
make use of the tag numbers far above the numbers of the standard FIX tag.

The most complex part of the development process was determining what tags in
FIX represented what message elements in ISO 15022 XML. When developing future
message translators for other message types this task would be made simpler by
convergence documentation which was to be released by ISO. This documentation
would describe mappings between FIX tags and message elements in ISO 15022
XML. The implementation of new message translators would probably still be time
consuming though.

A part of ISO 15022 XML that made application writing easier was the use of
XML. Many tools were available for parsing and validating XML messages. The use
of XML also made the messages relatively easy to understand for a human reader.
By comparison, the contents of a FIX message was very hard to understand without
having the standard documents at hand. However, this improved readability came
with a drawback. Using a human readable format made the messages longer and
less space efficient as can be seen in table 7.1 in chapter 7.
8.4 Recommendations for Future Work

At the time of writing this master’s thesis, not much of ISO 15022 XML had been released. The only part that was available was a draft of the indication of interest message. Due to the lack of material the work on the translator between ISO 15022 XML and FIX could not be completed. Only one example message translator, the Indication of Interest message translator, could be implemented. When more parts of the protocol are released it would be interesting to incorporate translators for more messages into the application. What was also needed in order to make the product ready for use was to test it more thoroughly and implement more logging facilities and error handling. One of the reasons the application was not thoroughly tested was that the draft version of the XML-schema released for the indication of interest message did not add many constraints on the design of the messages. It was thus not very useful when validating the translated messages.

Since no information about session level messages in ISO 15022 XML was released, no session handling facilities were implemented in the ISO 15022 XML engine. When further development of the product takes place the session handling is a part which will need some work.

A message translator for the message type indication of interest was implemented in the master’s project. This translator could translate most FIX tags into message elements and components in ISO 15022 XML. However, since it was hard to find the correct translation for some FIX tags and the draft for the indication of interest message was somewhat ambiguous, a few tags were not handled by the application. When selling the application as a product of its own it might be necessary to translate all tags of a FIX message. Translations for the tags not currently supported by the message translator would then have to be implemented with the help of the convergence documentation released by ISO.

When more parts of ISO 15022 XML are released it might be interesting to follow one of the other strategies presented in chapter 5. Implementing the other strategies would probably not be very hard since much of the code developed in this project could be reused.
References

[1] Century 24 Solutions,  


[3] FIX Protocol Limited,  

[4] FIX Protocol Limited,  
Version 4.2 with Errata 20010501.

[5] FIX Protocol Limited,  
*FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX)*, August 2001.  
Version 4.3.

[6] FIX Protocol Limited,  
URL: http://www.fixprotocol.org/cgi-bin/PrintPage.cgi?file=WORKGROUPS/994459967/SWIFTFIX_Rel_20010706.htm.  

[7] ISO/TC68/SC4,  

[8] ISO/TC68/SC4/WG10,  
URL: http://www.fixprotocol.org/cgi-bin/PrintPage.cgi?file=WORKGROUPS/994459967/WG10_FAQ_v1.22.htm.  

URL: http://www.fixprotocol.org/WORKGROUPS/994459967/
ISO15022_PreTrade_MessageIOL_draft.doc.

[11] OMG,

[12] PwC Consulting,
URL: www.sia.com/stp/pdf/Standards_Sub-Committee_Report_V2.0.PDF.

[13] Soliddata,

[14] SWIFT,
*SWIFT*, October 2002.

[15] W3C,
URL: http://www.w3c.org.
Appendix A

Class Diagram of the Developed Application