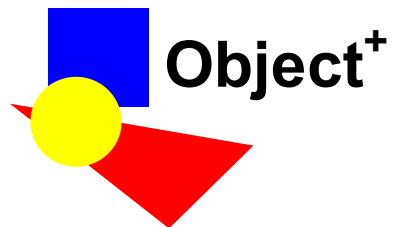


# THE UNIVERSAL FIX GATEWAY



- *General description of the Universal FIX Gateway concept*
- *Technical appendix*

May 2002  
Copyright © 2000-2002, Object+

# THE UNIVERSAL FIX GATEWAY

## Background

Over the years, Object+ has realized numerous connections between the software systems of its customers and electronic trading systems of exchanges and brokers. Such a connection offers functionality like entering orders in the electronic exchange, or monitor the market.

Creating such a connection involves not only establishing a *technical* link between the software modules, but also a *functional* link, in which the business model of the electronic exchange is mapped on the business model of the in-house applications. The business model defines the flow of messages a system uses to enter and confirm orders, to subscribe to market data, and so on. Furthermore, the business model defines the data exchanged in such transactions.

Designing and implementing business model mappings for every combination of electronic exchange and in-house application is an inefficient and error-prone approach. To do this in a much more reliable and cost-effective way, Object+ developed the concept of the Universal FIX Gateway.

## What is the Universal FIX Gateway

The basic concept in the Universal FIX Gateway is that the business models of different electronic trading systems are mapped on one “universal” business model. The universal business model in turn can then be used by in-house applications, which need not be aware of the specifics of the original electronic trading system. Figure 1 shows this in a schematic way.

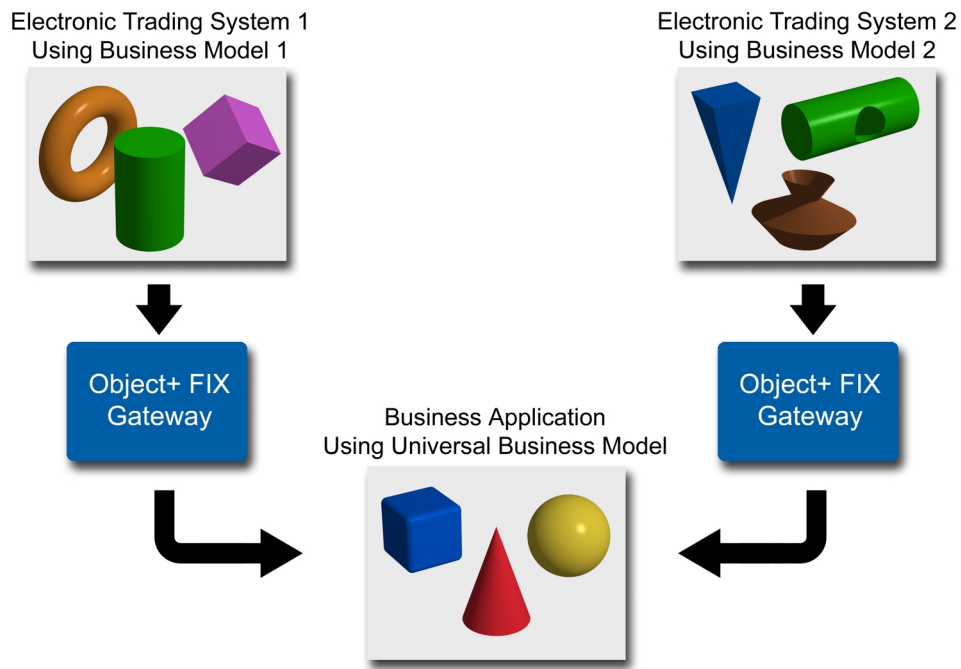


Figure 1

The translation between the universal business model and the model specific for the electronic exchange is taken care of by a translator module, designed and implemented by Object+.

The universal business model adopted by Object+ is the model defined by the industry-standard FIX (Financial Exchange) protocol. This widely accepted standard defines things like the types of orders that can be entered, the flow of events when an order is accepted (or rejected), the way an application receives market data updates, and so on.

The advantage of using the FIX standard is that software designers can take advantage of readily available resources like documentation, Internet discussion forums and testing tools.

A potential drawback is that some electronic exchanges offer more functionality than is accounted for in FIX. It is the Object+ point of view that the use of the universal business model should never hamper a software application, and therefore the Universal FIX Gateway concept allows for the extension of the FIX protocol in a well-organized way.

## Functionality provided by the Universal FIX Gateway translators

### Introduction

In order to actually use the Universal FIX Gateway concept, Object+ created a set of “translator” modules. Each of these translators is able to map the business model of a specific electronic trading system to the universal business model used by Object+. Apart from this primary function each translator supports a number of supporting, or secondary, functions. This is described in more detail below.

### Primary functions

The FIX standard, as defined by the FIX committee, can deal with several kinds of products: derivatives, equities, fixed income and collective investment vehicles.

The functionality defined by FIX are categorized as pre-trade, trade and post-trade functions.

The trade functions are:

- Order entry, modification and deletion, including multileg orders
- Receiving order confirmations
- Receiving transaction execution reports

The pre-trade functions are:

- Obtaining real-time market data
- Obtaining quotes
- Obtaining security definitions
- Obtaining security status
- Obtaining trading session status
- Event communication (news and e-mail)
- Advertisements and indication of interest

Finally the post-trade functions deal with:

- Allocations
- Settlements

- Trade capture (“street side”) reporting

Not all of these functions and product types are supported by every electronic trading system. And not all functions supported by electronic trading systems are defined in FIX. Schematically, this can be represented as two intersecting collections:

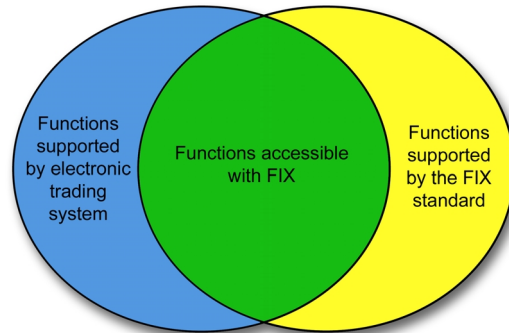


Figure 2

The blue collection represents the functions supported by a given electronic trading system. The yellow collection represents the functions supported by FIX. So the green part, the intersection, is the part of the functionality reachable with standard FIX messages. When Object+ designs and builds a FIX Gateway translator, it first analyzes which part of the blue section should also be covered in the translator. Subsequently, extensions to the FIX protocol are designed for this purpose. Part of the Object+ Universal FIX Gateway concept is a system to design and categorize these extensions; refer to “Extending the FIX business model” in the technical appendix for a more in-depth explanation on this subject.

After the interface design step, the software is designed and built to actually implement the interfaces. Normally, not *all* functions supported by the (extended) FIX interface are implemented; a subset is defined that covers the needs of the customers. So the functions actually supported in a given FIX Gateway translator can be represented as follows:

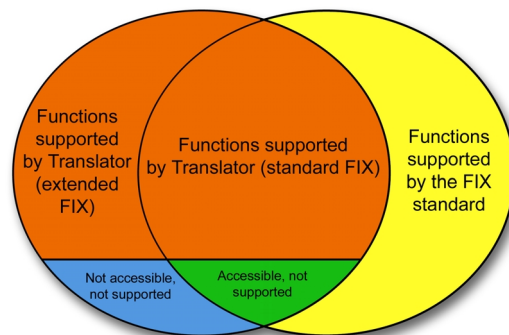


Figure 3

Here the orange collection is what is actually provided by a given FIX Gateway Translator. If required, the orange area can be extended to cover the entire blue area (in other words: everything offered by the electronic exchange). Because the interfaces have

been defined beforehand, these extensions can be implemented later on without any risk for already existing business applications.

### **Secondary functions**

Apart from the primary mapping functions as described, every FIX Gateway translator supports secondary functions, necessary for reliable deployment in a production environment.

- Error detection and recovery. The basic rule is that no run time error must go undetected. Examples of run time errors are a business application entering an incorrect order, but also unexpected loss of a data communication link or a lack of mass storage capacity. The translator makes sure every problem detected is either automatically solved, or is reported to the client software.
- Audit trails in the form of logging and tracing. Several configuration options can be set to trace exactly what data is exchanged between the business application and the translator, and between the translator and the electronic trading system. These traces are written to a text file that can be examined later on.
- Fail over. A FIX Gateway translator can be configured in such a way that persistent data is written to two locations: the primary and the fail over location. This technique allows the software process to be switched to a secondary computer in case of an emergency.

# THE UNIVERSAL FIX GATEWAY

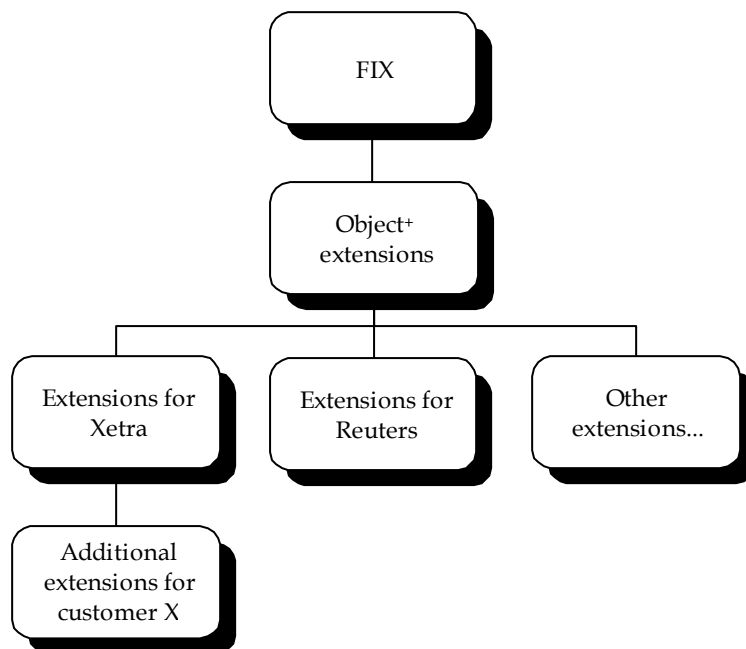
## TECHNICAL APPENDIX

### Extending the FIX business model

Theoretically, the FIX standard is designed to allow for extension: messages and data fields can be added in ways described by the standard. However, extending the standard should be done in a well-considered way. A careless approach can easily lead to compatibility problems in already existing software. After all, the business model is part of the interface exposed to client software, and the only way to guarantee backward compatibility is to offer a stable and consistent interface.

The Universal FIX Gateway organizes extensions to the FIX standard in multiple categories, the so-called “domains” (see figure 4). At the top of the drawing is the FIX domain, which represents the business model that follows the official FIX specifications to the letter. An extension to the FIX domain is the Object+ domain: it contains a number of messages and data fields that have a very generic purpose yet are missing in FIX. If possible, Object+ designs its extensions to the FIX standard in such a way that they fit in the Object+ domain, and therefore are available to all client programs.

If an interface must be designed (or extended) that is only suitable for one specific electronic trading system, it is categorized in a specific domain (like the Xetra domain) that is an extension to the Object+ domain.



**Figure 4**

This way the FIX extensions fit into a “domain taxonomy”. The general rule is: when a FIX Gateway can be used with messages from a specific domain (like “Xetra”) it can also be used with messages belonging to higher domains (Object+ and FIX). Every domain in the taxonomy (except for the top domain) is an extension of a more generic domain. So

when for example an application program wants to work with the Xetra translator module, and it does not want to use any Xetra specific or Object+ specific functionality, it can address the translator using standard FIX messages. When it wants to benefit from the Object+ extensions it uses the Object+ domain, and when it needs Xetra-specific functionality it must use features of the Xetra domain. This way an application programmer is always aware of the usability of the program in different environments. An interface from the Xetra domain will obviously never work on Reuters, but an interface from the Object+ domain will work on both. Furthermore the taxonomy helps to ensure backward compatibility, because a message defined within a specific domain will never change.

Figure 4 also shows the possibility for a custom-specific extension. This can be useful when a customer wants the translator to perform very specific functionality, tailor-made to be used by his in-house applications. In such a case a FIX-like interface is designed, and placed in accustom-specific domain.

### **Using a FIX Gateway Translator**

The FIX Gateway Translator is a separate application program that interacts with a business program. The Translator performs the role of server; the business program performs the role of client. The two communicate via a TCP connection to exchange FIX messages. Object+ developed a robust library, called FIXAPI, which is used at the client and server side to take care of all the complicated details of FIX message exchange, like encoding / decoding, session level handling, syntactic message validation and message transfer over TCP. FIXAPI is an integral part of the Universal FIX Gateway. By handling all technical FIX details automatically, the business application can concentrate on the real business issues.

The Gateway Translator is able to access a specific electronic trading system, and to that purpose might support messages and data that are extensions to the FIX standard. A client program can easily find out exactly what interfaces and implementations the Translator supports, by sending a special message called "Gestalt", which is part of the Object+ extensions. When the Translator receives this message, it replies with a message containing a list of the Translators capabilities. This way a client application can find out in an early stage if it connected to the right Translator (or to the correct version of the Translator).

The Translator program can generally be addressed with messages from different "domains", as explained in "Extending the FIX business model". It is up to the client program to decide which of the extensions it is going to use, if any. Extensions can take the form of completely new message types, or fields added to existing FIX messages, or even an extended use of a particular FIX data field. The client and server program must of course mutually agree on the meaning of the messages exchanged to guarantee error-free processing. To this purpose, Object+ added one (optional) data field to every message, the "Gateway Interface ID", or "GIID" for short. This is a number that denotes the domain the message at hand belongs to. When no GIID data is included in a message it is automatically considered a standard FIX message. The GIID allows a very safe exchange of messages, and prevents miscommunications. When the Translator receives a message containing the GIID of a specific domain (for example the "Xetra" specific domain), it runs additional checks on conditionally required fields in the message, and also checks for superfluous fields that are not allowed in the context of this domain.

When the message fails these validations, it is rejected. Experience shows it is better to reject such a message than to just ignore the superfluous fields and continue, because probably the client programmer had a specific reason for setting up the values of the “superfluous” fields.

The Translator uses the GIID included in the first message received (the Logon message) to determine the “preferred domain” of the client program. When for example the client includes the “Xetra” GIID in its Logon message, the Translator makes sure it supplies any Xetra specific data later on when the client subscribes to market data. When a client program logged on in the generic FIX way (without specifying a GIID), the Translator knows it must just supply the standard FIX data, and not the additional Xetra data, because it is probably dealing with a 100% standard FIX client program.

## 10 frequently asked questions about the Object+ FIX Gateway

1. *Does the Universal FIX Gateway give access to functions that are not in FIX, but are offered by a specific electronic trading system?*

Yes. If required the interface is extended by Object+ (in ways allowed by the FIX standard) to make these functions accessible.

2. *How can Object+ claim to support truly universal FIX while at the same time making extensions to it?*

The fact is that the interface exposed by the Gateway consists of a truly universal core, and optional extensions. The core functions of the interface are fully compliant with the official FIX standard, and the extensions are designed in the most generic way possible. Each Gateway provides a way for the business application to query what extended interfaces are supported and which functions are actually implemented. So if the business application only uses the core functions, it can remain fully unaware of the specific features of the trading system it connects to. If it *does* need the system-specific features, it can use the extensions in an orderly manner by first finding out (at run time) what interfaces are actually supported.

3. *Suppose I have written one business application that connects, via the Gateway, to the Eurex exchange. I want to connect the same application to Instinet. What steps would be involved?*

If your application only uses core functions, you must adapt some minor issues, like the user id and password provided in the Logon. Probably 99% of the application can be used straight away. When extended functions are used, the program will have to be adjusted for use with Instinet. However, as the application can easily find out at run time what extensions are supported by the Gateway, you can still end up with one program that uses either trading system.

4. *How do you deal with functions not offered by an electronic trading system? Do you still try to implement them in the Gateway?*

If the function can be added we normally do. For example, if a trading system offers a real time trade stream, but does not offer a function to query historic trades, we add a trade data store in the Gateway, in order to fully implement the FIX “Street Side Trade Capture Reporting” function.

5. *What happens if the electronic trading system releases a new version of its connection software?*

In such a case Object+ releases a new version of the Gateway, which takes care of any changes in the mapping between FIX and the electronic trading system. So the business applications that use the Gateway need no maintenance, unless it uses trading system specific data that is affected by the upgrade (for example when an exchange introduces a new type of product).

6. *What happens if the FIX committee releases a new version of the protocol, in which for example features that used to be an extension are now part of the “official” standard?*

In that case a Gateway will be released that supports the new FIX standard, but will remain backward compatible with the “old” interface. So newly developed business applications can be based on the new FIX standard, while existing applications continue to run without change.

7. *Must the business application take care of FIX message encoding and decoding?*

No, the details of FIX message encoding / decoding, as well as all FIX session level handling, syntactic message validation and message transfer are dealt with in the

Object+ FIXAPI library, which comes with is part of the Gateway. The business application can concentrate on the real business issues.

8. *Can a business application connect to multiple Gateways at the same time?*

Yes, this is fully supported by FIXAPI.

9. *What platforms and development environments are supported?*

The Gateway Translator is an application that can be deployed on Microsoft Windows or Sun Solaris (note the electronic trading system might impose additional requirements). It communicates with the business application over a TCP/IP connection. The business application can either be developed in C++ (for Windows or Sun Solaris) or in Visual Basic 6.0 (Windows only).

10. *What is the performance penalty imposed by the Gateway (as opposed to connecting to the electronic trading system directly)?*

It is obvious that the translation between FIX and the native model of the trading system consumes system resources, mainly CPU time. The overall message throughput depends on the capacity of the machine and other processes running on the same machine, but experience shows that several thousands of messages can be handled per second, with a very low latency per message.