
FIX Protocol basics

Prof. Dr. Bernd Ulmann

18-OCT-2010

Hochschule fuer Oekonomie und Management, Frankfurt

Introduction

What is the FIX Protocol?

- Acronym for *Financial Information eXchange* Protocol (not to be confused with *IPFIX*, the *Internet Protocol Flow Information Export*).
- Open standard which defines a message format as well as a communication model.
- This standard has been created by an industry consortium consisting of banks, independent vendors etc.
- The intended audience consists of financial institutions like banks, brokers, dealers, exchanges etc.
- FIX is *the* defacto standard in financial information exchange today.
- FIX is platform independent (concerning systems and networks).
- Central point for all information about FIX is <http://www.fixprotocol.org>.

How has FIX evolved over the years¹?

- FIX was conceived by the equity trading departments of *Fidelity Management & Research* and *Salomon Brothers*.
- After the feasibility of the concept developed by these two companies became apparent, a FIX committee was established which began work in June 1994.
- In 1995 the first Technical Committee Meeting took place.
- Also in January 1995, FIX was released to the financial community in version 2.7.
- In September 1995 version 3.0 of the FIX Protocol was released.

¹These slides are based on

<http://fixprotocol.org/implementation-guide/introduction.shtml>,
28.10.2010.

- The London FIX General Conference was held in June 1996.
- FIX 4.0 was released in 1997.
- FIX 4.1 was released in 1998.
- In June 1998 the FIX Committee structure was formalized.
- In March 1999 the Japanese FIX Commiettee was formalized.
- In April 1999 the FIX Procotol Ltd. was started.
- The FIX Protocol was extended for fixed income trading by *Putnam Investments* and *Merrill Lynch*. A pilot implementation was running in March 2000.
- FIX 4.2 was released in March 2000 and included four additional tags for fixed income trading.

- The extension also included so called *user defined tags*, *UDTs* for short.
- In 2000 the FIX Protocol organization established the *Fixed Income Working Group*, *FIWG* for short which became known as the *Global Fixed Income Committee (GFIC)*.
- The *FIX Protocol Limited (FPL)* entered so called *Statements of Understanding (SoU)* with other organizations like *SWIFT*, the *Bond Market Association* etc.

Who uses the FIX Protocol? The main user groups are these:

Buy-side firms: Communication with sell-side firms by means of pre-trade, trade and post-trade messages.

Sell-side firms: Communication with buy-side firms via pre-trade, trade and post-trade messages. In addition to that communication with exchanges and OTC markets in general.

Exchanges: Receiving trades from their members, sending execution reports etc. back to them.

Currently a wide variety of product classes are supported ranging from equities to fixed income products, derivatives and the like.

FIX offers a wide range of benefits in various areas:

- FIX simplifies the implementation of interfaces by using so called *FIX engines* (there are open source implementations as well as commercial ones).
- FIX increases the efficiency on the dealer side by saving time for price and execution data transmission.
- Since FIX defines message semantics, it reduces the risk of human error in trade entry etc. and makes it possible to detect errors earlier in the business process flow if they occur.
- FIX engines implement all necessary means for logging FIX message, for dealing with dropped connections etc. thus simplifying application development.
- Nearly every broker/dealer/exchange *speaks* FIX.
- Nearly all vendors of *order management systems*, *OMS*, offer FIX connectivity.

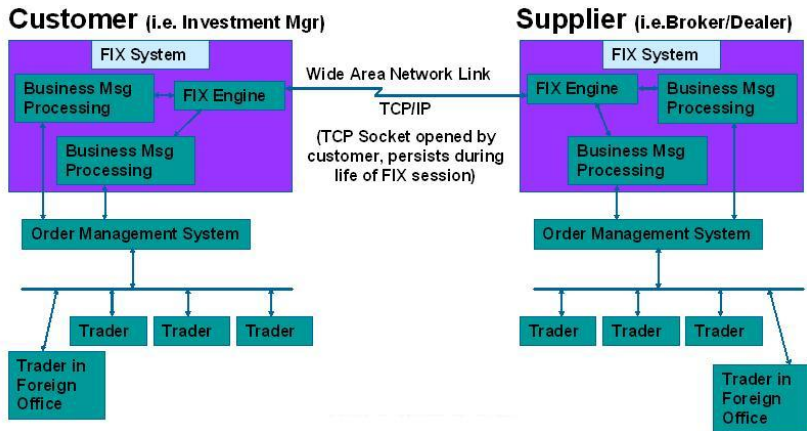
The main benefits of FIX from a trader's perspective are the following ones²:

- The use of FIX *"has allowed trading volumes to increase without corresponding increases in staffing"*.
- FIX *"allows traders to focus on dialogue and not on clerical tasks"*.
- FIX *"allows an individual to trade more stocks and decrease the chance of errors at the same time"*.
- *"With the increased volatility in the marketplace, FIX enables me to stay on top of my positions in real-time."*
- FIX *"prevents Nick Leeson-type activities before they can even happen, not just next day on confirmation or on settlement day"*.

²Cf. [Hong Kong 00][p. 25 ff.].

A typical FIX setup is shown below³:

FIX System Connectivity



³Cf. <http://upload.wikimedia.org/wikipedia/en/e/ea/Fix.jpg>, 28.10.2010.

Message structure and fields

Although FIX messages always have a similar structure, the number and type of possible fields depends strongly on the version of the FIX Protocol which is used.

At the time of this writing the latest version is FIX 5.0 although most current production systems use older versions 4.x (4.4 being the last).

FIX messages are grouped into two categories:

Admin messages: Connection establishment and termination, heartbeat messages etc. (logon, logoff, heartbeat, test request, resend request, reject, sequence reset, ...)

Application messages: Business related message transferring trade data etc. (advertisement, indication of interest, news, execution report, order cancel, new order, quote and many, many more)

Every FIX message is built according to the following structure:

Message header: Contains the message type, length,
sender/receiver name, sequence number, time stamp,
...

Message body: Contains session/application specific data.

Message trailer: Contains a message checksum and an optional signature.

The message consists of so called *FIX fields* which look like this:

`<tag>=<value><delimiter>`

The tag is a numerical identifier, the value is a string (which must be of correct type and length for this particular tag) and the delimiter is SOH (*Start of header* – ASCII 0x01) which is written as ^ in printed messages.

Although FIX supports literally hundreds of predefined tags, there is room for custom defined tags (field numbers 5000 to 9999).

The following example of a *New Order Single* message is taken from [FIXML 00][p. 8]:

```
8=FIX.4.1^9=0235^35=D^34=10^43=N^49=VENDOR^50=CUSTOMER^
56=BROKER^52=19980930-09:25:58^1=XQCCFUND^11=10^21=1^
55=EK^48=277461109^22=1^54=1^38=10000^40=2^44=76.750000^
59=0^10=165
```

Header: 8 (version), 9 (body length), 35 (MsgType), 34 (MsgSeqNum), 43 (PossDupFlag), 49 (SenderCompID), 115 (OnBehalfOfCompID), 56 (TargetCompID), 52 (time stamp)

Body: 1 (Account), 11 (ClOrdID), 21 (HandInst), 55 (Symbol), 48 (SecurityID), 22 (IDSource), 54 (Side), 38 (OrderQty), 49 (OrdType), 44 (Price), 59 (TimelnForce)

Trailer: 10 (Checksum)

The FIX Protocol supports many different message types – some of these are shown in the following:

- 0: Heartbeat
- 1: Test request
- 2: Resend request
- 3: Reject
- 4: Sequence reset
- 5: Logout
- 6: Indication of interest
- 7: Advertisement
- 8: Execution report
- 9: Order cancel reject
- A: Logon
- B: News
- C: Email
- D: Order single
- E: Order list

The very same message may also be formatted using FIXML (unfortunately FIXML is not in widespread use)⁴:

```
<?xml version='1.0'?><!DOCTYPE FIXML SYSTEM 'fixmlmain.dtd'>
<FIXML>
  <FIXMLMessage>
    <Header>
      ...
    </Header>
    <ApplicationMessage>
      <Order>
        <ClOrdID>12345</ClOrdID>
        <HandlInst Value="1"/>
        <Instrument>
          <Security>
            <Symbol>EK</Symbol>
          </Security>
        </Instrument>
        <Side Value="1"/>
        <OrderQuantity>
          <OrderQty>10000</OrderQty>
        </OrderQuantity>
        <OrderType>
          <MarketOrder OrdType="1"/>
        </OrderType>
        <Currency Value="USD"/>
      </Order>
    </ApplicationMessage>
  </FIXMLMessage>
</FIXML>
```

⁴Cf. [FIXML 00][p. 10].

FIXML can even be embedded into *standard* FIX messages⁵ to facilitate migration and maintain backward compatibility:

```
...  
49=BROKER  
56=HUB  
128=INST  
212=245  
213=<FIXML>  
    <Header>  
        ...  
    </Header>  
    <Indication>  
        ...  
    </Indication>  
</FIXML>  
...
```

⁵Cf. [FIXML 00][p. 12].

+:

- FIXML can be parsed with any XML parser.
- FIXML allows automatic message validation.
- FIXML is more human readable compared with traditional FIX.
- FIXML can be processed by various middlewares etc.

-:

- FIXML needs much more bandwidth than the traditional FIX Protocol.
- Parsing (and especially validating) FIXML takes way more CPU resources than processing traditional FIX messages.
- Still only a few parties use FIXML.

- [Shaik 10] contains a wealth of practical examples of FIX messages ranging from single orders to order cancel/replace etc.
- Some example heartbeat and IOI FIX messages can be found at <http://fixprotocol.org/specifications/TechResources-Examples>

Network

Communication using the FIX Protocol has the following properties:

- It is session based and thus point to point from a logical perspective.
- Communication partners act according to one of these roles:
 - Initiator:** Initiates a communication by sending a *logon* message and ends it with a *logout* message – this is the *client*.
 - Acceptor:** Receives a login request, validates it and establishes the connection – this is the *server*.
- The FIX protocol implements a session layer.
- Messages are identified by sequence numbers. These are used for resend or reject requests and the like.
- Normally new sessions start with sequence number 1.

FIX is rather network agnostic in that it supports a wide variety of network models. Basically there are four network architectures to choose from:

Leased line: In this case a leased line is employed between two counter parts – the big advantage is that this line is truly private. On the other hand this solution becomes quickly expensive when many leased lines and/or high data rates/low latencies are necessary.

Internet: This is the cheapest solution from a technical point of view. The main disadvantages are stability, bandwidth and latency issues – all of these parameters can and will vary extremely. Another disadvantage is the need for encryption (SSL, AES etc.).

P2P-VPN: Quite like the Internet variant but the encryption is normally done at the router level.

Hub-and-Spoke: Here a central FIX engine will be employed in the

FAST

FAST is the acronym for *FIX Adapted for STraming* and describes a protocol which has been developed for the one-way exchange of data between a sender and one or multiple receivers.

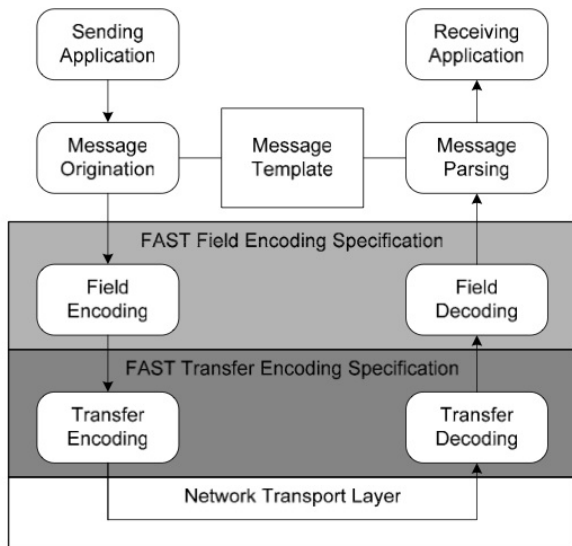
To quote [FAST 06][p. 5 f.]:

- *"FAST, at its core, is a data compression algorithm which when properly implemented will significantly reduce bandwidth requirements and latency between sender and receiver."*
- *"FAST ist an extension of the base FIX specification [...]. FAST exists as a stand alone specification which can be used within either broadcast or point-to-point transports."*

FAST performs the compression at two levels:

- Field encoding (controlled by templates which specify the message structure)
- Transfer encoding

The following picture shows the levels FAST operates on⁶:



⁶Cf. [FAST 06][p. 6].

- A typical area for using FAST is broadcast communication where one sender distributes data to a multitude of receivers.
- This is normally implemented using so called *multicasts* and UDP as the transport mechanism.
- Since UDP is a connection less communication model it is important for FAST to be employed on a frame-by-frame basis, thus data sent and received should never span multiple frames.

FIX engines

An actual implementation of the FIX Protocol is called a *FIX engine*. There is a wide variety of FIX engines available on the market – some of which are open source while the majority are rather expensive and sometimes only usable with other systems from a particular vendor.

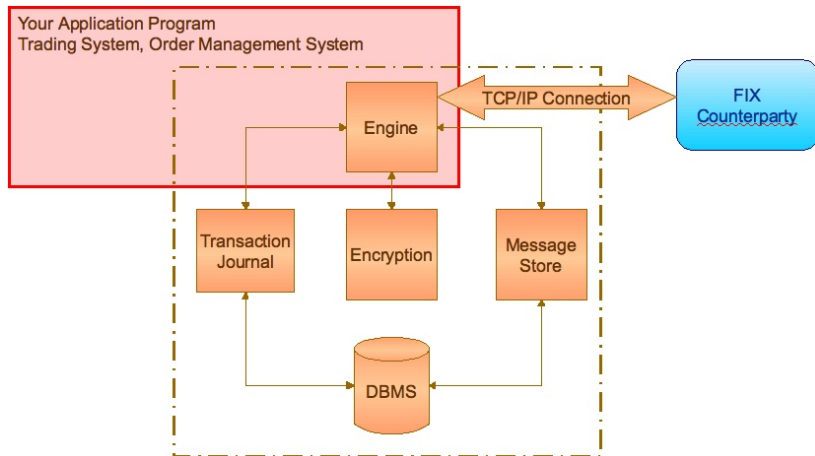
For small applications it might even be an option to develop your own FIX engine – the protocol is quite simple and if you can live without semantics checking, writing your own FIX engine could result in a remarkable short time to market (the author would, of course, use Perl for the implementation of such a small FIX engine).

Selecting a FIX engine is not a simple task and involves looking at the following criteria⁷ at least:

⁷Cf. <http://www.fixprotocol.org/implementation-guide/engine.shtml>, 28.10.2010.

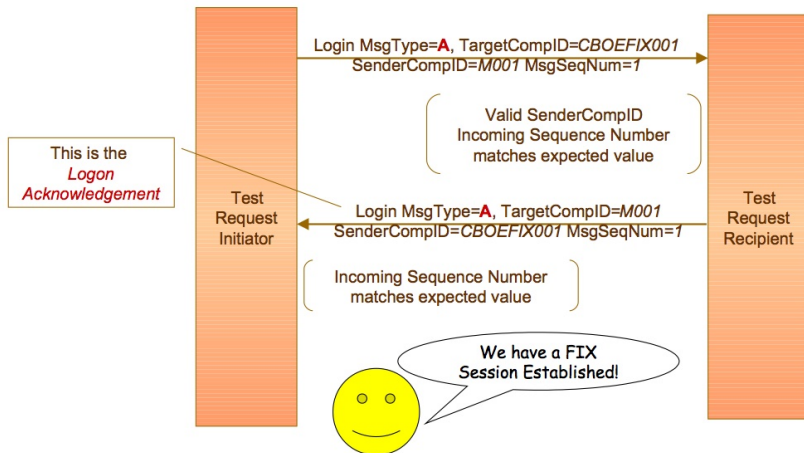
- What FIX versions are supported?
- Are all tags you need supported?
- Is the FIX engine able to run multiple FIX Protocol versions?
- Does it support your asset class?
- How about stability, extendability etc.?
- How much throughput is possible? Which memory/CPU requirements does the engine have?
- Does the engine run in your preferred hardware/software environment (this includes necessary databases, too)?
- Do you need a FIX engine that is capable of taking care of business logic or will you implement the business logic for your connection(s) at the application level?
- Does the vendor of your business system offer an own FIX engine?
- What about support? Are the sources available?
- Are there monitoring utilities included?

The following picture⁸ shows the anatomy of a typical FIX engine:



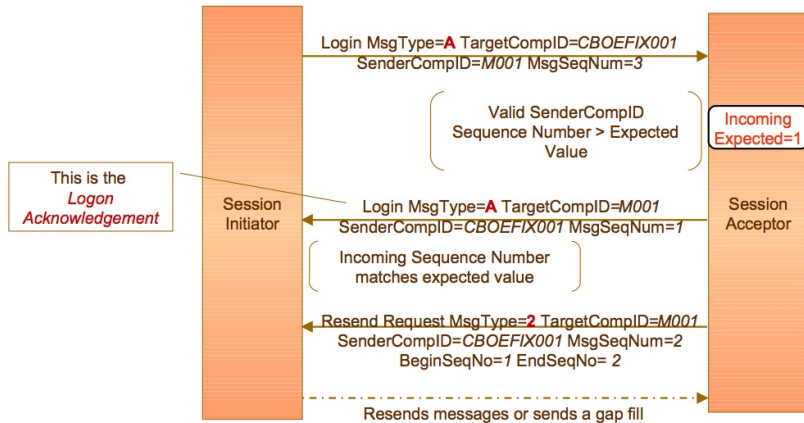
⁸Cf. [Northey 04][p. 8].

A FIX engine implements the FIX session level protocol and takes care of things like logon, log out, sequence numbering, message resend requests etc. A typical logon scenario looks like this⁹:



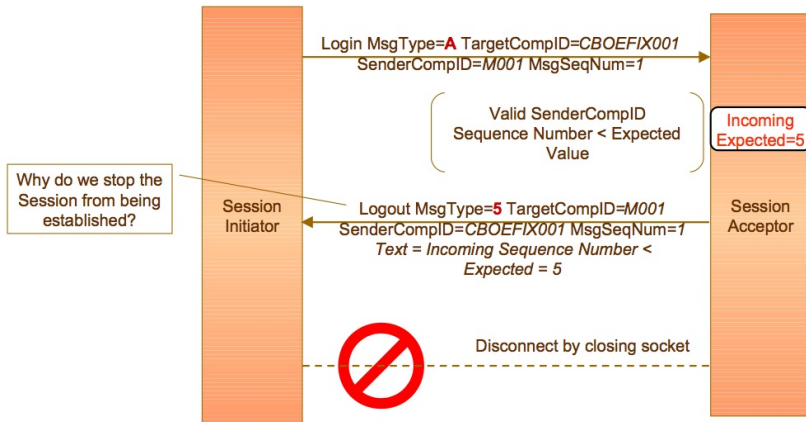
⁹Cf. [Northey 04][p. 12].

At logon time the sequence number is checked - if it is too high, something has been lost and has to be resent etc.¹⁰:



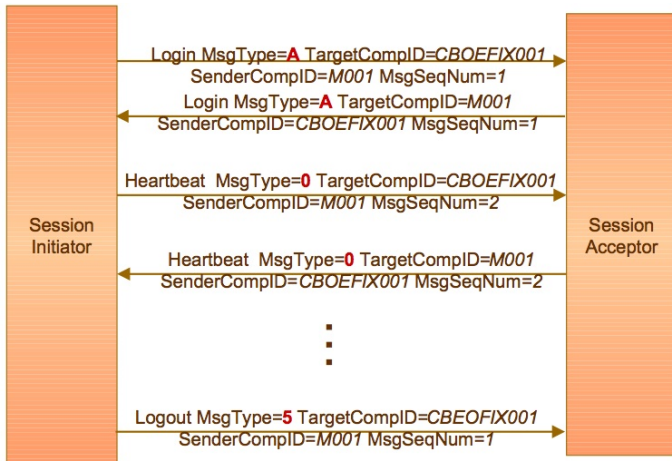
¹⁰Cf. [Northey 04][p. 16].

Dealing with a sequence number being too low results in closing the connection¹¹:



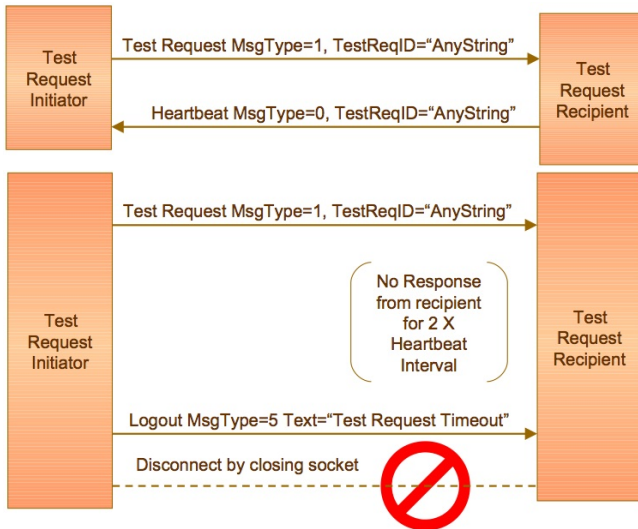
¹¹Cf. [Northey 04][p. 17].

During an established session heartbeat packets are exchanged in regular (definable) intervals to make sure that the connection is still valid¹²:



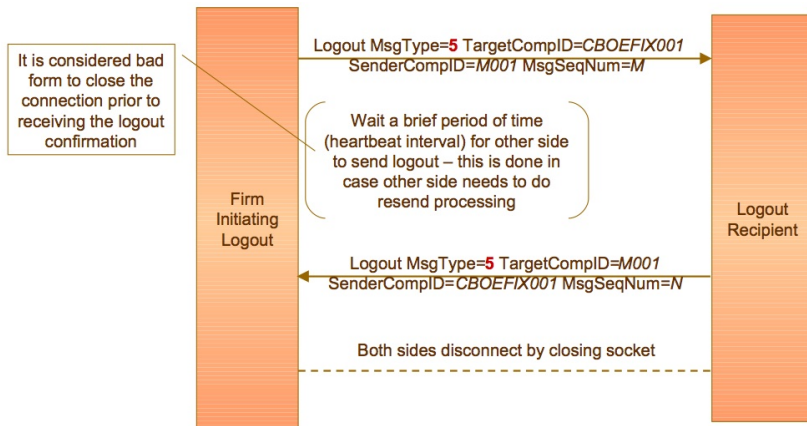
¹²Cf. [Northey 04][p. 20].

In addition to this test requests are supported¹³:



¹³Cf. [Northey 04][p. 23 f.].

Handling a logout request works like this¹⁴:



¹⁴Cf. [Northey 04][p. 27].

The following list of FIX engines is far from being complete – nearly every software vendor in the financial sector has its own FIX engine – the following entries just serve as examples:

FIX Antenna C++:

http://b2bits.com/trading_solutions/fix_engine_cpp.html
(also available for Java and .NET)

QuickFIX: A free production quality FIX engine implementation (see <http://www.quickfixengine.org/>).

QuickFIX/J: Quite the same as QuickFIX but completely Java based (see <http://www.quickfixj.org/>).

VersaFix: An open source .NET based FIX engine, implemented in C# (see <http://sourceforge.net/projects/versafix/>).

UL FIX: Yet another free FIX implementation (cf. http://www.ullink.com/index.php?page=free_fix_engine).

After you have chosen a FIX engine to be used and integrated it into your existing environment, you have to test it prior to use it in a production environment.

Most communication partners like Bloomberg etc. require the following tests to be performed successfully with their support before you will be allowed to communicate with them via FIX:

Connectivity testing: Basic connectivity, reconnect in case of errors etc. ($\sim 10\%$)

Session level testing: Are the generated messages valid and compatible with your communication partner etc. ($\sim 10\%$)

In addition to that you will need more tests inhouse:

FIX/OMS (*order management system*) tests: $\sim 40\%$

Integration testing: $\sim 40\%$

Session level tests normally include the following scenarios¹⁵:

- Stopping heartbeat on client and/or host to simulate a flaky connection.
- Send messages with sequence numbers being too high and too low and check the response of the FIX engines.
- Check how the FIX engines deal with messages they have missed (or think they have missed).
- ...

¹⁵Cf. [Johnson, Rhodes 01][p. 32].

Typical test scenarios from the business level viewpoint are the following¹⁶:

Orders:

- Test all required parameters and their domains (numerical values, strings containing special characters, date/time values, ...).
- The same tests are required for optional parameters.
- Test optional order types like Stop or Stop Limit etc.

Cancels:

- Simple cancel.
- Cancel after a Partially Filled.
- Partially filled while pending cancel.
- Unsolicited cancels etc.

¹⁶Cf. [Johnson, Rhodes 01][p. 30].

FIX tools

FIXForge FIX Dictionary:

<http://www.onixs.biz/tools/fixdictionary/>

FIXimate FIX Dictionary: This is the ultimate resource for digging deeper into FIX message of various versions (see <http://www.fixprotocol.org/FIXimate3.0/>).


FIXopaedia: <http://btobits.com/fixopaedia/index.html>

FIXwiki: <http://fixwiki.org/fixwiki/FIXwiki>

Mini-FIX: This is a handy Windows tool which implements a simple FIX client/server with a simple GUI. Mini-FIX has been proven to be very valuable during development of FIX based interfaces etc. (see <http://69.64.38.175/>).

ValidFIX: This is a free online tool for inspecting FIX messages as well as logs produced by FIX applications (see <http://www.validfix.com>).

[About Us](#) | [Contact Us](#) | [Site Map](#) | [News](#) | [Home](#)




[Home](#) | [selfConnect Products](#) | [directConnect Solutions](#) | **[Tools](#)** | [Support](#) | [Downloads](#)

Browse FIX Dictionary:				
	Messages by		Fields by	
FIX 4.0	MsgType	Name	Tag	Name
FIX 4.1	MsgType	Name	Tag	Name
FIX 4.2	MsgType	Name	Tag	Name
FIX 4.3	MsgType	Name	Tag	Name
FIX 4.4	MsgType	Name	Tag	Name
FIX 5.0	MsgType	Name	Tag	Name
FIX 5.0.SP1	MsgType	Name	Tag	Name
FIX 5.0.SP2	MsgType	Name	Tag	Name

Note: You must have **JavaScript enabled** in your web-browser to use all dictionary features.

For more information on **Onix Solutions** or any of our products please e-mail info@onixs.biz.



The Onixs FIX Dictionary

is based on the latest specifications of **FIX Protocol**.

We would be glad to improve the FIX Dictionary on the base of your recommendations: please contact us at support@onixs.biz

Onixs FIX Dictionary is an integral part of **Onix Solutions FIX/FAST Messaging Suite**:

- [Onixs .NET FIX Engine](#)
- [Onixs C++ FIX Engine](#)
- [Onixs Java FIX Engine](#)
- [CME FAST Handler](#)
- [ICAP EBS FAST Handler](#)
- [FIX Analyser](#)



FIXimateSM v3.0

About FIXimate

Disclaimer

Release Notes

Copyright and Usage Policy

Download standalone Fiximate

Select Fix Version

FIX.4.4 -

FIX.4.4 - English

Specification Home Page

Message by type:

8

Go

Field by tag:

11

Go

Field by name:

Go

Field by regex:

Go

match: ^=start, \$=end, .=any

Datatypes

Fields

Components

Message Summary

Application Level Messages

Pre Trade

Trade

Post Trade

Other

Session Layer

Messages

Components

FIX.4.4 Message

ExecutionReport [type '8']

<ExecRpt>

The execution report message is used to:

1. Confirm the receipt of an order
2. Confirm changes to an existing order (i.e. accept cancel and replace requests)
3. Relay order status information
4. Relay fill information on working orders
5. Relay fill information on tradeable or restricted tradeable quotes
6. Reject orders
7. Report post-trade fees calculations associated with a trade

Tag or Component	Field Name	FIXML name	Req'd	Comments	Depr.
Component	StandardHeader		✓	MsgType = 8	
37	OrderID	@OrdID	✓	OrderID is required to be unique for each chain of orders.	
198	SecondaryOrderID	@OrdID2		Can be used to provide order id used by exchange or executing system.	
526	SecondaryClOrdID	@ID2			
527	SecondaryExecID	@ExecID2			
11	ClOrdID	@ID		Required for executions against electronically submitted orders which were assigned an ID by the institution or intermediary. Not required for orders manually entered by the broker or fund manager (for CIV orders).	
41	OrigClOrdID	@OrigID		Conditionally required for response to an electronic Cancel or Replace request (ExecType=PendingCancel, Replace, or Cancel). ClOrdID of the previous accepted order (NOT the initial order of the day) when canceling or replacing an order.	
583	ClOrdLinkID	@LnkID			
693	QuoteRespID	@RspID		Required if responding to a QuoteResponse message. Echo back the Initiator's value specified in the message.	
790	OrdStatusReqID	@StatReqID		Required if responding to and if provided on the Order Status Request message. Echo back the value provided by the requester.	
584	MassStatusReqID	@ReqID		Required if responding to a Order Mass Status Request. Echo back the value provided by the requester.	
911	TotNumReports	@TotNumRpts		Can be used when responding to an Order Mass Status Request to identify the total number of Execution Reports which will be returned.	
912	LastRptRequested	@LastRptReced		Can be used when responding to an Order Mass Status Request to indicate that this is the last Execution Reports which will be returned as a result of the request.	
Component	Parties	Pty		Insert here the set of "Parties" (firm identification) fields defined in "Common Components of Application Messages"	
229	TradeOriginationDate	@OrigDt			



Preconfigured software for connectivity to major exchanges, trading networks and broker dealers worldwide

Site [FIXopaedia](#)



[B2BITS](#)

[SOLUTIONS](#)

[SERVICES](#)

[SUPPORT](#)

[PERFORMANCE LAB](#)

[CONTACT US](#)

[Home](#) / [Solutions](#) / [FIXopaedia](#)

FIXopaedia



+1 (888) 378-0666



sales@b2bits.com

Welcome to FIXopaedia the fastest, most comprehensive and feature rich FIX reference source. B2BITS' FIXopaedia is built on top of our complete set of FIX Dictionaries and allows examining the elements required for session and application level messages across all versions of the FIX Protocol (FIX 4.0 - 5.0, 5.0 SP1, 5.0 SP2):

- Component Blocks
- Message Name
- Message Type
- Field Name
- Field Type

Please click below to see the full logical specification of every message and tag type for the version of FIX that you require.

Version 2.2.1

- [FIX 4.0](#)
- [FIX 4.1 \(with errata 19990630\)](#)
- [FIX 4.2 \(with errata 20010501\)](#)
- [FIX 4.3 \(with errata 20020920\)](#)
- [FIX 4.4 \(with errata 20030618\)](#)
- [FIX 5.0](#)
- [FIX 5.0 SP1](#)
- [FIX 5.0 SP2](#)
- [FIXT 1.1](#)

The following FIX engines provide complete support of all these dictionaries:



[FIX Antenna C++](#)



[FIX Antenna .NET](#)



[FIX Antenna Java](#)



[CME FIX/FAST Market Data Adaptor](#)

Next Generation FIX Logs Analyzers



[FIXEye for Windows](#)



[FIXGrep for Windows](#)



[FIXGrep for Linux](#)

Performance Benchmarks




[Check Out the Latest Performance Benchmarks](#)

Compare



[B2BITS® FIX Products vs. Open Source FIX Solutions](#)



**CAMERON
EDGE**

navigation

- Main Page
- FIX Specification
- Recent changes
- Random page
- FIX Spec Errors
- FIX Repo Errors
- FIX Spell Checking

search

Go Search

toolbox

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

page discussion view source history

FIXwiki

FIXwiki is a [wiki](#) containing data from the [FIX specification](#). There is a FIXwiki page for each FIX message, component, field, value or type. In addition to definitions taken from the specification, each page also has an area for user comments, clarifications, corrections, examples or suggestions. Please contribute. FIXwiki was created by John Cameron of [Cameron Edge](#) for the benefit of the FIX community. It is completely free.

Quick Start

Log in then type one of the following into the search box on the left:

- a FIX message or component name (eg Confirmation or Parties)
- a FIX field name (eg Quantity)
- a FIX tag number (eg 54)


or if you just want to browse, click on one of the following links:

- [FIX Messages](#)
- [FIX Components](#)
- [FIX Fields](#)
- [FIX Values](#)
- [FIX Types](#)

Click on the Cameron Edge logo at the top left at any time to return to this main page.


More Detail

[Using FIXwiki.](#)
[Structure of FIXwiki.](#)
[About FIXwiki.](#)



**CAMERON
EDGE**

This page was last modified on 26 August 2009, at 00:54. This page has been accessed 10,461 times. Content is available under [FIXwiki copyrights](#). [About FIXwiki](#)



The Mini-FIX application window displays a list of FIX messages in the left pane and a detailed view of a selected message in the right pane.

Message List (Left Pane):

- 8=FIX.4.2;9=60;35=0;34=217;49=EXECUTOR;52=20091006-20:34:16;57;356=CLIENT1;110=153
- 8=FIX.4.2;9=69;35=1;34=218;49=EXECUTOR;52=20091006-20:34:18;57;456=CLIENT1;1112=TEST;110=185
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=11;452=20091006-20:35:41;1112=TEST;110=222
- 8=FIX.4.2;9=60;35=0;34=219;49=EXECUTOR;52=20091006-20:34:28;58;156=CLIENT1;110=157
- 8=FIX.4.2;9=69;35=1;34=220;49=EXECUTOR;52=20091006-20:34:30;58;156=CLIENT1;1112=TEST;110=170
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=11;562=20091006-20:35:53;112=TEST;110=225
- 8=FIX.4.2;9=60;35=0;34=221;49=EXECUTOR;52=20091006-20:34:40;58;156=CLIENT1;110=144
- 8=FIX.4.2;9=69;35=1;34=222;49=EXECUTOR;52=20091006-20:34:42;58;256=CLIENT1;1112=TEST;110=178
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=11;652=20091006-20:36:05;112=TEST;110=225
- 8=FIX.4.2;9=127;35=D;49=CLIENT1;56=EXECUTOR;34=11;752=20091006-20:36:10;11=999;21=155=ERICB.ST;154=1160=200
- 8=FIX.4.2;9=156;35=B;34=223;49=EXECUTOR;52=20091006-20:36:47;53;156=CLIENT1;116=100;11=999;14=100;117=1020=0
- 8=FIX.4.2;9=112;35=D;49=CLIENT1;56=EXECUTOR;34=11;852=20091006-20:36:15;11=999;121=155=ERICA.ST;154=1160=200
- 8=FIX.4.2;9=124;35=D;4=224;49=EXECUTOR;52=20091006-20:36:53;105=CLIENT1;11858=Conditionally Required Field
- 8=FIX.4.2;9=60;35=0;34=225;49=EXECUTOR;52=20091006-20:35:03;58;156=CLIENT1;110=155
- 8=FIX.4.2;9=69;35=1;34=226;49=EXECUTOR;52=20091006-20:35:05;58;156=CLIENT1;1112=TEST;110=186
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=11;952=20091006-20:36:28;112=TEST;110=223
- 8=FIX.4.2;9=60;35=0;34=227;49=EXECUTOR;52=20091006-20:35:15;53;256=CLIENT1;110=155
- 8=FIX.4.2;9=69;35=1;34=228;49=EXECUTOR;52=20091006-20:35:17;59;156=CLIENT1;1112=TEST;110=188
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=12;052=20091006-20:36:40;112=TEST;110=219
- 8=FIX.4.2;9=60;35=0;34=229;49=EXECUTOR;52=20091006-20:35:27;59;156=CLIENT1;110=162
- 8=FIX.4.2;9=69;35=1;34=230;49=EXECUTOR;52=20091006-20:35:29;59;156=CLIENT1;1112=TEST;110=185
- 8=FIX.4.2;9=65;35=0;49=CLIENT1;56=EXECUTOR;34=12;152=20091006-20:36:52;112=TEST;110=223
- 8=FIX.4.2;9=60;35=0;34=231;49=EXECUTOR;52=20091006-20:35:39;58;156=CLIENT1;110=162
- 8=FIX.4.2;9=127;35=D;49=CLIENT1;56=EXECUTOR;34=12;252=20091006-20:37:03;11=999;21=155=ERICB.ST;154=1160=200
- 8=FIX.4.2;9=156;35=B;34=232;49=EXECUTOR;52=20091006-20:35:40;89;156=CLIENT1;116=100;11=999;214=100;117=1120=0

Message Detail View (Right Pane):

Tag	Name	Content	Meaning
8	BeginString	FIX.4.2	
9	BodyLength	156	
35	MsgType	8	
34	MsgSeqNum	232	
49	SenderCompID	EXECUTOR	
52	SendingTime	20091006-20:35:40...	
56	TargetCompID	CLIENT1	
6	AvgPx	100	
11	ClOrdID	9992	
14	CumQty	1000	
17	ExecID	11	
20	ExecTransType	0	NEW
31	LastPx	100	
32	LastShares	1000	
37	OrderID	11	
38	OrderQty	1000	
39	OrdStatus	2	FILLED
54	Side	1	BUY
55	Symbol	ERICB.ST	
150	ExecType	2	FILL
151	LeavesQty	0	
10	Checksum	071	

Connected

Thank you for your interest.

The author would also like to thank Dr. Reinhard Steffens for his support and proof reading.

Bibliography



[FAST 06] *A Basic User's Guide to Implementing The FAST Protocol*, Version 1.0, January, 2006, <http://www.fixprotocol.org/documents/2301/A%20Basic%20Guide%20to%20FAST%20v1.0.pdf>, requested October 31th, 2010



[FIXML 00] *FIX AND XML: FIXML*, <http://www.fixprotocol.org/documents/646/FIXSession-New.pdf>, requested 28.10.2010



[Hong Kong 00] *Financial Information eXchange – General Conference*, Hong Kong, March 30, 2000, <http://fixprotocol.org/documents/627/>



[Johnson, Rhodes 01] Sam Johnson, David Rhodes, *FIX/FIXML implementation*



[Northey 04] Jim Northey, *Introduction to FIX – The FIX Session Layer*, http://fixprotocol.org/documents/742/FIX_Session_Layer_rev1.ppt, requested Oct. 31th, 2010



[Shaik 10] Khader Shaik, *FIX Protocol One Day Course*, http://www.ksvali.com/wp-content/uploads/2009/02/fix_1day_allsections.pdf, requested 28.10.2010