FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX)

Version 4.0

January 10, 1997

DISCLAIMER

THE INFORMATION CONTAINED HEREIN AND THE FINANCIAL INFORMATION EXCHANGE PROTOCOL (COLLECTIVELY, THE "FIX PROTOCOL") ARE PROVIDED "AS IS" AND NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL MAKES ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, AS TO THE FIX PROTOCOL (OR THE RESULTS TO BE OBTAINED BY THE USE THEREOF) OR ANY OTHER MATTER AND EACH SUCH PERSON AND ENTITY SPECIFICALLY DISCLAIMS ANY WARRANTY OF ORIGINALITY, ACCURACY, COMPLETENESS, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SUCH PERSONS AND ENTITIES DO NOT WARRANT THAT THE FIX PROTOCOL WILL CONFORM TO ANY DESCRIPTION THEREOF OR BE FREE OF ERRORS. THE ENTIRE RISK OF ANY USE OF THE FIX PROTOCOL IS ASSUMED BY THE USER.

NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL SHALL HAVE ANY LIABILITY FOR DAMAGES OF ANY KIND ARISING IN ANY MANNER OUT OF OR IN CONNECTION WITH ANY USER'S USE OF (OR ANY INABILITY TO USE) THE FIX PROTOCOL, WHETHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL (INCLUDING, WITHOUT LIMITATION, LOSS OF DATA, LOSS OF USE, CLAIMS OF THIRD PARTIES OR LOST PROFITS OR REVENUES OR OTHER ECONOMIC LOSS), WHETHER IN TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), CONTRACT OR OTHERWISE, WHETHER OR NOT ANY SUCH PERSON OR ENTITY HAS BEEN ADVISED OF, OR OTHERWISE MIGHT HAVE ANTICIPATED THE POSSIBILITY OF, SUCH DAMAGES.

No proprietary or ownership interest of any kind is granted with respect to the FIX Protocol (or any rights therein).

PREFACE

The Financial Interface eXchange (FIX) effort was initiated in 1992 by a group of institutions and brokers interested in streamlining their trading processes. These firms felt that they, and the industry as a whole, could benefit from efficiencies derived through the electronic communication of indications, orders and executions. The result is FIX, an open message standard controlled by no single entity, that can be structured to match the business requirements of each firm. The benefits are:

* From the business flow perspective, FIX provides institutions and brokers a means of reducing the clutter of unnecessary telephone calls and scraps of paper, and facilitates targeting high quality information to specific individuals.
* For technologists, FIX provides an open standard that leverages the development effort so that they can efficiently create links with a wide range of counter-parties.
* For vendors, FIX provides ready access to the industry, with the incumbent reduction in marketing effort and increase in potential client base.

Openness has been the key to FIX' success. For that reason, while encouraging vendors to participate with the standard, FIX has remained vendor neutral. Similarly, FIX avoids over-standardization. It does not demand a single type of carrier (e.g., it will work with leased lines, frame relay, Internet, etc.), nor a single security protocol. It leaves many of these decisions to the individual firms that are using it. We do expect that, over time, the rules of engagement in these non-standardized areas will converge as technologies mature.

FIX is now used by a variety of firms and vendors. It has clearly emerged as the inter-firm messaging protocol of choice. Periodic Technical Committee meetings are held to discuss modification of the specification and are open for all to attend. Those interested in providing input to the protocol are encouraged to contact the Technical Committee Chairpersons, Bob Lamoureux, Fidelity Management and Research Company, 617-563-6648 (blam@fmrco.com) or Chris Morstatt, Salomon Brothers Inc, 212-783-2865 (cmorstatt@sbi.com). Technical Committee meetings are announced via email and on the WWW, go to http://world.std.com/~fix for details.

We look forward to your participation.

The FIX Committee

December, 1996

Alliance Capital Management
American Century Investments
CS First Boston
Fidelity Capital Markets
Fidelity Management and Research
Goldman Sachs & Co.
Morgan Stanley & Co.
Paine Webber
Putnam Investments
Salomon Brothers Inc
Scudder, Stevens and Clark, Inc
State Street Global Advisors

Contents

FINANCIAL INFORMATION EXCHANGE PROTOCOL

INTRODUCTION

The Financial Information Exchange (FIX) Protocol is a message standard developed to facilitate the electronic exchange of information related to securities transactions. It is intended for use between trading partners wishing to automate communications.

The message protocol, as defined, will support a variety of business functions. FIX was originally defined for use in supporting US domestic equity trading with message traffic flowing directly between principals. As the protocol evolved, a number of fields were added to support limited cross-border and fixed income trading. Similarly, the protocol was expanded to allow third parties to participate in the delivery of messages between trading partners. As subsequent versions of FIX are released it is expected that functionality will continue to expand.

FIX was written to be independent of any specific communications protocol (X.25, asynch, TCP/IP, etc.) or physical medium (copper, fiber, satellite, etc.) chosen for electronic data delivery.

The protocol is defined at two levels; session and application. The session level is concerned with the delivery of data while the application level defines business related data content. This document is organized to reflect the distinction.

FIX MESSAGE FORMAT AND DELIVERY

The following section summarizes general specifications for constructing and transmitting FIX messages.

Message Format

The general format of a FIX message is a standard header followed by the message body fields and terminated with a standard trailer.

Each message is constructed of a stream of <tag>=<value> fields.

Except where noted, fields within a message can be defined in any sequence (i.e. relative position of a field within a record is inconsequential); exceptions are explicitly defined otherwise (certain header fields and fields within repeating data groups).

It is permissible for fields to be repeated. In the case where a field allows multiple values, these repeating fields are logically added together to form the data for that field. It is also possible for a field to be contained in both the clear text portion and the encrypted data sections of the same message. This is normally used for validation and verification. For example, sending the *SenderCompID* in the encrypted data section can be used as a rudimentary validation technique. In the cases where the clear text data differs from the encrypted data, the encrypted data should be be considered more reliable. (A security warning should be generated).

All fields in a FIX message are terminated by a delimiter character; the non-printing, ASCII "SOH" (#001), is used for this purpose. Records are delimited by the "SOH" character following the CheckSum field; all records begin with the "8=FIX.x.y" string and terminate with "10=nnn<SOH>".

There shall be no embedded delimiter characters within fields except for data type *data*.

Data Types:

Data types are mapped to ASCII strings as follows:

- int: Sequence of digits without commas or decimals and optional sign character (ASCII characters "**-**" and "**0**" - "**9**"). The sign character utilizes one byte (i.e. positive int is "**99999**" while negative int is "**-99999**").

  Examples: 723 in field 21 would be mapped int as |**21=723**|.

-723 in field 12 would be mapped int as |**12=-723**|

- float: Sequence of digits with optional decimal point and sign character (ASCII characters "**-**", "**0**" - "**9**" and "**.**"); the absence of the decimal point within the string will be interpreted as the float representation of an integer value. All float fields must accommodate up to fifteen significant digits.
- char: Alpha-numeric free format strings, can include any character or punctuation except the delimiter. All char fields are case sensitive (i.e. **morstatt ≠ Morstatt**).
- time: Time/date combination in YYYYMMDD-HH:MM:SS format, colons and dash required. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-59.
- date: Date in YYYYMMDD format. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31.
- data: Raw data with no format or content restrictions. Data fields are always immediately preceded by a length field. *Caution: may contain the delimiter (SOH) character.*

Sequence Numbers:

All FIX messages are identified by a unique sequence number. Sequence numbers are initialized at the start of each FIX session (see Session Protocol section) starting at 1 (one) and increment throughout the session. Monitoring sequence numbers will enable parties to identify and react to missed messages and to gracefully synchronize applications when reconnecting during a FIX session.

Each session will establish an independent incoming and outgoing sequence series; participants will maintain a sequence series to assign to outgoing messages and a separate series to monitor for sequence gaps on incoming messages.

Heartbeats:

During periods of message inactivity FIX applications will generate *Heartbeat* messages at regular time intervals. The heartbeat is useful for monitoring the status of the communication link and to identify incoming sequence number gaps. The heartbeat interval is declared by the session initiator using the HeartBtInt field in the *Logon* message. The heartbeat interval timer should be reset after every message is transmitted (not just heartbeats).

Ordered Message Processing:

The FIX protocol assumes complete ordered delivery of messages between parties. Implementors should consider this when designing message gap fill processes. Two options exist for dealing with gaps, either request all messages subsequent to the last message received or ask for the specific message missed while maintaining an ordered list of all newer messages. For example if the receiver misses the second of five messages sent, messages 3-5 can be ignored and a resend request generated for messages 2-5 or messages 3-5 can be saved and only message 2 resent. In both cases, messages 3-5 should not be processed before message 2.

Possible Duplicates:

When a FIX engine is unsure if a message was successfully received at its intended destination or when responding to a resend request a possible duplicate (*poss dupe*) message is generated. The poss dupe will be a retransmission (with the same sequence number) of the application data in question with the PossDupFlag included and set to "Y" in the header. It is the receiving application's responsibility to handle the poss dupe message (i.e. treat as a new message or discard as appropriate). All messages created as the result of a resend request will contain the PossDupFlag field set to “Y”, messages lacking the PossDupFlag field or with the PossDupFlag field set to “N” should be treated as original transmissions. *Note: When retransmitting a message with the PossDupFlag set to Y, it is always*

necessary to recalculate the CheckSum value. The only fields that can change in a Poss Dup message are the CheckSum, OrigSendingTime, SendingTime, BodyLenth and PossDupFlag. Fields related to encryption (SecureDataLen and SecureData) may also require recasting.

Possible Resends:

Application level messages which appear to the end-user to be in an ambiguous state may be resent with the PossResend flag set. An example of when this is useful is the case where an order remains unacknowledged for an inordinate length of time and the end-user suspects it had never been sent. The receiving application must recognize this flag and interrogate internal fields (order number, etc.) to determine if this order has been previously received. *Note: the Poss Resend message will contain exactly the same body data but will have the PossResend flag and will have a new sequence number. In addition the CheckSum field will require recalculation and fields related to encryption (SecureDataLen and SecureData) may also require recasting.*

Data Integrity:

FIX allows the integrity of message data content to be verified in two ways; verification of record length and a simple checksum of characters.

The record length is indicated in the BodyLength field and is verified by counting the number of characters in the message following the BodyLength field up to, and including, the delimiter immediately preceding the CheckSum tag ("10=").

The CheckSum integrity check is calculated by summing the binary value of each character from the "8" of "8=" up to and including the <SOH> character immediately preceding the CheckSum tag field and comparing the least significant eight bits of the calculated value to the CheckSum value (see Appendix B for a complete description).

Required Fields:

Each message within the protocol is comprised of *required*, *optional* and *conditionally required* (fields which are required based on the presence or value of other fields) fields. Systems should be designed to operate when only the required and conditionally required fields are present.

Message Acknowledgment:

The FIX session protocol is based on an optimistic model; normal delivery of data is assumed (i.e. no acknowledgment of individual messages) with errors in delivery identified by message sequence number gaps. Each is identified by a unique sequence number. It is the receiving application's responsibility to monitor incoming sequence numbers to identify message gaps for response with resend request messages.

The FIX protocol does not support individual message acknowledgment. However, a number of application messages require explicit application level acceptance or rejection. Orders, cancel requests, cancel/replace requests and allocation require specific application level response, executions can be rejected with the DK message but do not require explicit acceptance.

Encryption:

The exchange of sensitive data across public carrier networks may make it advisable to employ data encryption techniques to mask the application messages.

The choice of encryption method will be determined by mutual agreement of the two parties involved in the connection..

Any field within a message can be encrypted and included in the SecureData field, however, certain explicitly identified fields must be transmitted unencrypted; the clear (unencrypted) fields can be repeated within the SecureData field to serve as an integrity check of the clear data.

When encryption is employed, it is recommended (however, not required) that all fields within the message body be encrypted.

Embedded in the protocol are fields which enable the implementation of a public key signature and encryption methodology, straight DES encryption and clear text. The encryption methodology is declared in the *Logon* message and should conform to that which was previously agreed. (For more detail on implementation of various encryption techniques see the application notes section on the FIX Home Page.)

User Defined Fields:

In order to provide maximum flexibility for its users, the FIX protocol accommodates *User Defined Fields.* These fields are intended to be implemented between consenting trading partners and should be used with caution to avoid conflicts which will arise as multiple parties begin implementation of the protocol. It is suggested that if trading partners find that particular User Defined Fields add value, they should be recommended to the FIX Technical Committee for inclusion in a future FIX version.

The tag numbers 5000 to 9999 have been reserved for use with user defined fields.

SESSION PROTOCOL

A FIX session is defined as a bi-directional stream of ordered messages between two parties within a continuous sequence number series. A single FIX session can exist across multiple physical connections; parties can connect and disconnect multiple times while maintaining a single FIX session. Connecting parties must bi-laterally agree as to when sessions are to be started/stopped based upon individual system and time zone requirements. It is recommended that a new FIX session be established once within each 24 hour period.

The FIX session protocol is based on an optimistic model; normal delivery of data is assumed (i.e. no communication level acknowledgment of individual messages) with errors in delivery identified by message sequence number gaps. This section provides details on the implementation of the FIX session layer and dealing with message sequence gaps.

NOTE: ***The session protocol has changed dramatically in FIX 4.0. The major differences are in the areas of message sequence number generation/control and in session initialization.***

The following terms are used throughout this section:

- **Valid FIX Message** is a message that is properly formed according to this specification and contains a valid body length and checksum field
- **Initiator** is the party who establishes the telecommunications link and initiates the session via transmission of the initial *Logon* message.
- **Acceptor** is the receiving party of the FIX session. This party has responsibility to perform first level authentication and formally declare the connection request “accepted” through transmission of an acknowledgment *Logon* message.

A FIX session is comprised of three parts: logon, message exchange and logout.

Logon -

Establishing a FIX connection involves three distinct operations: creation of a telecommunications level link, authentication/acceptance of the initiator by the acceptor and message synchronization (initialization). The sequence of connection follows:

- The session initiator establishes a telecommunication link with the session acceptor.
- The initiator sends a *Logon* message. The acceptor will authenticate the identity of the initiator by examining the *Logon* message. The *Logon* message will contain the data necessary to support the authentication method previously agreed upon between the parties. If the initiator is successfully authenticated, the acceptor responds with a *Logon* message, if authentication fails, the session acceptor should shut down the connection. The session initiator may begin to send messages immediately following the *Logon* message, however, the session may not be supported by the other side at this time. The initiator must wait for the confirming *Logon* message from the acceptor before declaring the session fully established.

  After the initiator has been authenticated, the acceptor will respond immediately with a confirming *Logon* message. Depending on the encryption method being used for that session, this *Logon* message may or may not contain the same session encryption key. The initiator side will use the *Logon* message being returned from the acceptor as confirmation that a FIX session has been established. If the session acceptor has chosen to change the session encryption key, the session initiator must send a third *Logon* back to the other side in order to acknowledge the key change request. This also allows the session acceptor to know when the session initiator has started to encrypt using the new session key. Both parties are responsible for infinite loop detection and prevention during this phase of the session.
- After authentication, the initiator and acceptor must synchronize their messages through interrogation of the *MsgSeqNo* field. A comparison of the *MsgSeqNo* in the *Logon* message to the internally monitored next expected sequence number will indicate any message gaps. Likewise, the initiator can detect gaps by comparing the acknowledgment *Logon* message *MsgSeqNo* to the next expected value. Refer to the section on message recovery later in this document for dealing with message gaps.

Message exchange -

After completion of the initialization process, normal message exchange begins. The formats for all valid messages are detailed in the sections 'Administrative Messages' and 'Application Messages'.

Logout -

Normal termination of the message exchange session will be completed via the exchange of *Logout* messages. Termination by other means should be considered an abnormal condition and dealt with as an error.

Before actually closing the session, the Logout initiator should wait for the opposite side to respond with a confirming Logout message. This gives the remote end a chance to perform any Gap Fill operations that may be necessary. The session may be terminated if the remote side does not respond in an appropriate timeframe.

Note: the process of logging out does not affect the state of any orders. All active orders will continue to be eligible for execution after logout.

Message Recovery -

During initialization, or in the midst of a FIX session, message gaps may occur which are detected via the tracking of incoming sequence numbers. The following section provides details on how to recover messages.

As previously stated, each FIX participant must maintain two sequence numbers for each FIX session, one each for incoming and outgoing messages which are initialized at 1 (one) at the beginning of the FIX session. Each message sent is assigned a unique (by connection) sequence number which is incremented after each message. Likewise, every message received has a unique sequence number and the incoming sequence counter is incremented after each message.

When the incoming sequence number does not match the expected number corrective processing is required. **If the incoming message has a sequence number less than expected and the PossDup flag is not set it indicates a serious error and it is strongly recommended that the session be terminated and manual intervention be initiated.** If the incoming sequence number is greater than expected it indicates that messages were missed and retransmission of the messages is requested via the *Resend Request* (see the earlier section, *Ordered Message Processing).*

Note: For the purposes of the following paragraphs *requester* refers to the party requesting the resend and *resender* refers to the party responding to the request. The process of resending and synchronizing messages is referred as "gap filling".

Upon receipt of a *Resend Request* the resender can respond in one of three ways:

1. retransmit the requested messages (in order) with the original sequence numbers and *PossDupFlag* set to "Y"
2. issue a *SeqReset-GapFill* message to replace the retransmission of administrative and application messages
3. issue a *SeqReset-Reset* to force sequence number synchronization

During the gap fill process, certain administrative messages should not be retransmitted. Instead, a special *SeqReset-GapFill* message is generated. The administrative messages which are not to be resent are: *Logon*, *Logout*, *ResendReq*, *HeartBeat*, *TestReq* and *SeqReset*. The *SeqReset-GapFill* can also be used to skip application messages that the sender chooses not to retransmit (e.g. aged orders).

All FIX implementations must monitor incoming messages to detect inadvertently retransmitted administrative messages (*PossDup* flag set indicating a resend). When received, these messages should be processed for sequence number integrity only; the business/application processing of these message should be skipped (e.g. do not initiate gap fill processing based on a resent *ResendReq)*.

If there are consecutive administrative messages to be resent, it is suggested that only one *SeqReset-GapFill* message be sent in their place. The sequence number of the *SeqReset-GapFill* message is the next expected outbound sequence number. The *NewSeqNum* field of the GapFill message itself contains the sequence number of the highest administrative message in this group plus 1. For example, during a Resend operation there are 7 administrative messages in a row waiting to be resent. They start with sequence number 9 and end with sequence number 15. Instead of transmitting 7 Gap Fill messages (which is perfectly legal, but not network friendly), a *SeqReset-GapFill* message may be sent. The sequence number of the Gap Fill message itself is set to 9 because the remote side is expecting that message next. The *NewSeqNum* field of the GapFill message contains the number 16, because that will be the sequence number of the next message to be transmitted.

Sequence number checking is a vital part of FIX session management. However, a discrepancy in the sequence number stream is handled differently for certain classes of FIX messages. The table below lists the actions to be taken in the case where the incoming sequence number is greater than the expected incoming sequence number.

NOTE: In *ALL* cases, the FIX session should be terminated if the incoming sequence number is less than expected and the PossDup flag is not set. A *Logout* message with some descriptive text should be sent to the other side before closing the session.

Response by Message Type

Message Type	Action to Be Taken on Sequence # mismatch
Logon	Must always be the first message transmitted. Authenticate and accept the connection. After sending a *Logon* confirmation back, send a *ResendRequest* if a message gap was detected in the *Logon* sequence number.
Logout	If a message gap was detected, issue a *ResendRequest* to retrieve all missing messages followed by a *Logout* message which serves as a confirmation of the logout request. **DO NOT** terminate the session. The initiator of the *Logout* sequence has responsibility to terminate the session. This allows the *Logout* initiator to respond to any *ResendRequest* generated. If this side was the initiator of the *Logout* sequence, then this is a *Logout* confirmation and the session should be immediately terminated upon receipt. The only exception to the "do not terminate the session" rule is for an invalid Logon attempt. The session acceptor has the right to send a Logout message and terminate the session immediately. This minimizes the threat of unauthorized connection attempts.
ResendRequest	Perform the Resend processing first, followed by a *ResendRequest* of your own in order to fill the incoming message gap.
SeqReset-Reset	Ignore the incoming sequence number. The *NewSeqNum* field of the *SeqReset* message will contain the sequence number of the next message to be transmitted.
SeqReset-GapFill	Send a *ResendRequest* back. Gap Fill messages behave similar to a *SeqReset* message. However, it is important to insure that no messages have been inadvertently skipped over. This means that *GapFill* messages must be received in sequence. An out of sequence *GapFill* is an abnormal condition
All Other Messages	Perform Gap Fill operations.

Message header

Each message, administrative or application, is preceded by a standard header. The header is used to identify the message type, length, destination, sequence number, origination point and time.

Two fields are provided for use when resending messages. The PossDupFlag is set when resending a message as the result of a session level event (i.e. the retransmission of a message reusing a sequence number). The PossResend is set when reissuing a message with a new sequence number (e.g. resending an order). The receiving application should process these messages as follows:

PossDup - if a message with this sequence number has been previously received, ignore message, if not, process normally.

PossResend - forward message to application and determine if previously received (i.e. verify order id and parameters)

The message header format is as follows:

Message Header

Tag	*Field Name*	*Req'd*	*Comments*
8	BeginString	Y	FIX.4.0 (*Always unencrypted, must be first field in message*)
9	BodyLength	Y	(*Always unencrypted, must be second field in message*)
35	MsgType	Y	(*Always unencrypted, must be third field in message*)
49	SenderCompID	Y	(*Always unencrypted*)
56	TargetCompID	Y	(*Always unencrypted*)
115	OnBehalfOfCompID	N	Trading partner company ID used when sending messages via a third party *(Can be embedded within encrypted data section.)*
128	DeliverToCompID	N	Trading partner company ID used when sending messages via a third party *(Can be embedded within encrypted data section.)*
90	SecureDataLen	N	Required to identify length of encrypted section of message. (*Always unencrypted*)
91	SecureData	N	Required when message body is encrypted. Always immediately follows SecureDataLen field.
34	MsgSeqNum	Y	*(Can be embedded within encrypted data section.)*
50	SenderSubID	N	*(Can be embedded within encrypted data section.)*
57	TargetSubID	N	"ADMIN" reserved for administrative messages not intended for a specific user. *(Can be embedded within encrypted data section.)*
116	OnBehalfOfSubID	N	Trading partner SubID used when delivering messages via a third party. *(Can be embedded within encrypted data section.)*
129	DeliverToSubID	N	Trading partner SubID used when delivering messages via a third party. *(Can be embedded within encrypted data section.)*
43	PossDupFlag	N	Always required for retransmissions, whether prompted by the sending system or as the result of a resend request. *(Can be embedded within encrypted data section.)*
97	PossResend	N	Required when message may be duplicate of another message sent under a different sequence number. *(Can be embedded within encrypted data section.)*
52	SendingTime	Y	*(Can be embedded within encrypted data section.)*
122	OrigSendingTime	N	Required for message resends. If data is not available set to same value as SendingTime *(Can be embedded within encrypted data section.)*

Message trailer

Each message, administrative or application, is terminated by a standard trailer. The trailer is used to segregate messages and contains the three digit character representation of the Checksum value.

The message trailer format is as follows:

Message Trailer

Tag	*Field Name*	*Req'd*	*Comments*
93	SignatureLength	N	Required when trailer contains signature. *Note: Not to be included within SecureData field*
89	Signature	N	*Note: Not to be included within SecureData field*
10	CheckSum	Y	(*Always unencrypted, always last field in message*)

ADMINISTRATIVE MESSAGES

The administrative class of messages are intended to address the utility needs of the protocol. The following section describes the use of each message and provides the message layout.

Administrative messages will be generated from both sides of the connection.

Heartbeat -

The Heartbeat is useful for monitoring the status of the communication link and to identify when the last of a string of messages was not received.

When either end of a FIX connection has not sent any data for [HeartBtInt] seconds, it will transmit a Heartbeat message. When either end of the connection has not received any data for (HeartBtInt + "some reasonable transmission time") seconds, it will transmit a Test Request message. If there is still no Heartbeat message received after (HeartBtInt + "some reasonable transmission time") seconds then the connection should be considered lost and corrective action be initiated. If HeartBtInt is set to zero then no regular heartbeat messages will be generated. Note that a test request message can still be sent independent of the value of the HeartBtInt which will force a Heartbeat message.

Heartbeats issued as the result of Test Request must contain the TestReqID transmitted in the Test Request message. This is useful to verify that the Heartbeat is the result of the Test Request and not as the result of a regular timeout.

The heartbeat format is as follows:

Heartbeat

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 0
112	TestReqID	N	Required when the heartbeat is the result of a Test Request message.
	Standard Trailer	Y	

Logon Message -

The logon message is utilized to authenticate a user attempting to establish a connection to a remote system. The logon message must be the first message sent by the application requesting to initiate a FIX session.

The HeartBtInt (108) field is used to declare the timeout interval for generating heartbeats.

Upon receipt of a Logon message the session acceptor will authenticate the party requesting connection and issue a Logon message as acknowledgment that the connection request has been accepted. The acknowledgment Logon can also be used by the initiator to validate that the connection was established with the correct party.

The session acceptor must be prepared to immediately begin processing messages after receipt of the Logon. The session initiator can choose to begin transmission of FIX messages before receipt of the confirmation Logon, however it is recommended that normal message delivery wait until after the return Logon is received to accommodate encryption key negotiation.

The confirmation Logon can be used for encryption key negotiation. If a session key is deemed to be weak, a stronger session key can be suggested by returning a Logon message with a new key. This is

only valid for encryption protocols that allow for key negotiation. (See the FIX Home Page Application notes for more information on a method for encryption and key passing.)

The logon format is as follows:

Logon

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = A
98	EncryptMethod	Y	(*Always unencrypted*)
108	HeartBtInt	Y	
95	RawDataLength	N	Required for some authentication methods
96	RawData	N	Required for some authentication methods
	Standard Trailer	Y	

Test Request -

The test request message is utilized to force a heartbeat from the opposing application. The test request message is useful for checking sequence numbers or verifying communication line status. The opposite application will respond to the Test Request with a Heartbeat containing the TestReqID.

The TestReqID is used to verify that the opposite application is generating the heartbeat as the result of Test Request and not a normal timeout. The opposite application will include the TestReqID in the resulting Heartbeat. Any string can be used as the TestReqID (one suggestion is to use a timestamp string).

The test request format is as follows:

Test Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 1
112	TestReqID	Y	
	Standard Trailer	Y	

Resend Request -

The resend request is sent by the receiving application to initiate the retransmission of messages. This function is utilized if a sequence number gap is detected, if the receiving application lost a message, or as a function of the initialization process.

The resend request can be used to request a single message, a range of messages or all messages subsequent to a particular message.

Note: the sending application may wish to consider the message type when resending messages; e.g. if a new order is in the resend series and a significant time period has elapsed since its original inception, the sender may not wish to retransmit the order given the potential for changed market

conditions. (The Sequence Reset-GapFill message is used to skip messages that a sender does not wish to resend.)

Note: it is imperative that the receiving application process messages in sequence order, e.g. if message number 7 is missed and 8-9 received, the application should ignore 8 and 9 and ask for a resend of 7-9.

- To request a single message: BeginSeqNo = EndSeqNo
- To request a range of messages: BeginSeqNo = first message of range, EndSeqNo = last message of range
- To request all messages subsequent to a particular message: BeginSeqNo = first message of range, EndSeqNo = 999999

The resend request format is as follows:

Resend Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 2
7	BeginSeqNo	Y	
16	EndSeqNo	Y	
	Standard Trailer	Y	

Reject -

The reject message should be issued when a message is received which cannot be passed through to the application level. An example of when a reject may be appropriate would be the receipt of a message with invalid basic data (e.g. MsgType=&) which successfully passes de-encryption, CheckSum and BodyLength checks. (As a rule, messages should be forwarded to the trading application for business level rejections whenever possible.)

Rejected messages should be logged and the incoming sequence number incremented.

Note: When a message is received which is garbled, cannot be parsed or fails a data integrity check, the receiving application should disregard the message. Processing of the next valid FIX message will cause detection of a sequence gap and a Resend Request will be generated. Logic should be included in the FIX engine to recognize the possible infinite resend loop which may be encountered in this situation.

Generation and receipt of a Reject message should be dealt with as an indication of a serious error by both parties as it may be the result of faulty logic in either the sending or receiving application.

If the sending application chooses to retransmit the rejected message it should be assigned a new sequence number.

Whenever possible, it is strongly recommeded that the cause of the failure be described in the Text field (e.g. INVALID DATA - FIELD 35).

The reject format is as follows:

Reject

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 3
45	RefSeqNo	Y	MsgSeqNum of rejected message
58	Text	N	Where possible, message to explain reason for rejection
	Standard Trailer	Y	

Sequence Reset (Gap Fill) -

The sequence reset message is used by the sending application to reset the incoming sequence number on the opposing side. The sequence reset message can be used in the following situations:

- During normal resend processing, the sending application may choose not to send a message (e.g. an aged order), the Sequence Reset can be used to mark the place of that message.
- During normal resend processing a number of administrative messages are not resent, the Sequence Reset message is used to fill the sequence gap created.
- In the event of an application failure it may be necessary to force synchronization of sequence numbers on the sending and receiving sides

The sending application will initiate the sequence reset. The message is used to reset the value of the next sequence number to be transmitted.

If the GapFill field is not present (or set to N), it can be assumed that the purpose of the sequence reset message is to recover from an out-of-sequence condition, therefore, the MsgSeqNum in the header should be ignored (i.e. the receipt of a sequence reset message with an out of sequence MsgSeqNum should not generate resend requests).

If the Gap Fill field is present (and equal to Y), the MsgSeqNum should conform to standard message sequencing rules.

The sequence reset can only increase the sequence number; if a sequence reset is received attempting to decrease the next expected sequence number the message should be rejected and treated as a serious error.

The sequence reset format is as follows:

Sequence Reset

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 4
123	GapFillFlag	N	
36	NewSeqNum	Y	
	Standard Trailer	Y	

Logout -

The logout message is to used to initiate or confirm the termination of a FIX session. Disconnection without the exchange of logout messages should be interpreted as an abnormal condition.

Before actually closing the session, the logout initiator should wait for the opposite side to respond with a confirming logout message. This gives the remote end a chance to perform any Gap Fill operations that may be necessary. The session may be terminated if the remote side does not respond in an appropriate timeframe.

The logout initiator should not send any messages after the logout.

The logout format is as follows:

Logout

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 5
58	Text	N	
	Standard Trailer	Y	

APPLICATION MESSAGES

The exchange of business related information is accomplished through the passing of application messages. The application message is composed of the standard header followed by the message body and trailer.

Descriptions and formats of the specific messages follow.

Advertisements -

Advertisement messages are used to announce completed transactions. The advertisement message can be transmitted in various transaction types; NEW, CANCEL and REPLACE. All message types other than NEW modify the state of a previously transmitted advertisement identified in AdvRefID.

The advertisement record format is as follows:

Advertisement

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 7
2	AdvID	Y	
5	AdvTransType	Y	
3	AdvRefID	N	Required for Cancel and Replace AdvTransType messages
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
4	AdvSide	Y	
53	Shares	Y	
44	Price	N	
15	Currency	N	Indication without currency field is interpreted as US dollars.
60	TransactTime	N	
58	Text	N	
	Standard Trailer	Y	

Indications of Interest -

Indication of interest messages are used to market merchandise which the broker is buying or selling in either a proprietary or agency capacity. The indications can be time bound with a specific expiration value. Indications are distributed with the understanding that other firms may react to the message first and that the merchandise may no longer be available due to prior trade.

Indication messages can be transmitted in various transaction types; NEW, CANCEL, and REPLACE. All message types other than NEW modify the state of the message identified in IOIRefID.

The indication of interest message format is as follows:

Indication of Interest

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 6
23	IOIid	Y	
28	IOITransType	Y	
26	IOIRefID	N	Required for Cancel and Replace IOITransType messages
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
54	Side	Y	Side of Indication Valid values: 1 = Buy 2 = Sell
27	IOIShares	Y	
44	Price	N	
15	Currency	N	Indication without currency field is interpreted as US dollars.
62	ValidUntilTime	N	
25	IOIQltyInd	N	
24	IOIOthSvc	N	Applicable only if advertised on public IOI service.
130	IOINaturalFlag	N	
104	IOIQualifier	N	
58	Text	N	
	Standard Trailer	Y	

News -

The news message is intended for use as a general free format message between the broker and institution. The message contains flags to identify the news item's urgency and to allow sorting by subject company (symbol). The News record can be originated at either the broker or institution side.

The news message format is as follows:

News

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = B
42	OrigTime	N	
61	Urgency	N	
46	***RelatdSym***	***N***	***Can be repeated multiple times if message is related to multiple symbols.***
33	LinesOfText	Y	
58	Text	Y	Repeating field, number of instances defined in LinesOfText
95	RawDataLength	N	
96	RawData	N	
	Standard Trailer	Y	

Email -

Format and purpose similar to News message, however, intended for private use between two parties.

The email message format is as follows:

Email

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = C
94	EmailType	Y	
42	OrigTime	N	
46	***RelatdSym***	***N***	***Can be repeated multiple times if message is related to multiple symbols.***.
37	OrderID	N	
11	ClOrdID	N	
33	LinesOfText	Y	
58	Text	Y	Repeating field, number of instances defined in LinesOfText
95	RawDataLength	N	
96	RawData	N	
	Standard Trailer	Y	

Quote Request -

In some markets it is the practice to request quotes from brokers prior to placement of an order. The quote request message is used for this purpose.

Quotes can be requested on specific securities or for forex rates.

Securities quotes can be requested as either market quotes or for a specific quantity and side. If OrderQty and Side are absent, a market-style quote (bid x offer, size x size) will be returned.

The symbol used for forex quotes is, in ISO codes, "currency1.currency2" (e.g. GBP.USD) and the quote will be returned as a rate expressed as currency1/currency2.

Forex quotes can be requested as indicative or at a specific quantity level. If an indicative quote is requested (OrderQty and Side are absent), the broker has discretion to quote at either a specific trade level and side or to provide an indicative quote at the mid-point of the spread. The broker can also choose to respond to an indicative quote by sending multiple quote messages specifying various levels and sides.

The quote request message format is as follows:

Quote Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = R
131	QuoteReqID	Y	
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
140	PrevClosePx	N	Useful for verifying security identification
54	Side	N	
38	OrderQty	N	
	Standard Trailer	Y	

Quote -

The quote message is used as the response to a quote request message and can be used to publish unsolicited quotes.

Quotes supplied as the result of a Quote Request message are tagged with the appropriate QuoteReqID, unsolicited quotes can be identified by the absence of a QuoteReqID.

The symbol used for forex quotes is, in ISO codes, "currency1.currency2" (e.g. GBP.USD) and the quote will be returned as a rate expressed as currency1/currency2. BidPx indicates the rate at which the broker is willing to buy currency1 and deliver currency2, OfferPx indicates the rate at which the broker is willing to sell currency1 and receive currency2. Indicative rates are quoted in the BidPx field and may contain a level in the BidSize field.

Orders can be generated based on Quotes, quoted orders include the QuoteID and are OrdType=Quoted Order.

The quote message format is as follows:

Quote

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = S
131	QuoteReqID	N	Required when quote is in response to a Quote Request message
117	QuoteID	Y	
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
132	BidPx	Y	
133	OfferPx	N	
134	BidSize	N	
135	OfferSize	N	
62	ValidUntilTime	N	
	Standard Trailer	Y	

New Order - Single -

The new order message type is used by institutions wishing to electronically submit securities and forex orders to a broker for execution.

Orders can be submitted with special handling instructions and execution instructions. Handling instructions refer to how the broker should handle the order on its trading floor (see HandInst field); execution instructions contain explicit directions as to how the order should be executed (see ExecInst field).

New Order messages received with the PossResend flag set in the header should be validated by ClOrdID and order parameters (side, symbol, quantity, etc.) to determine if the order had been previously submitted. PossResends previously received should be acknowledged back to the client via an Execution - Status message. PossResends not previously received should be processed as a new order and acknowledged via an Execution - New message.

To support forex accommodation trades, two fields, ForexReq and SettlCurrency, are included in the message. To request a broker to execute a forex trade in conjunction with the securities trade, the institution would set the ForexReq = Y and SettlCurrency = "intended settlement currency". The

broker would then execute a forex trade from the execution currency to the settlement currency and report the results via the execution message in the NetMonies and SettlCurr fields.

The order message can also be used to request a straight forex trade. Conventions for identifying a forex transaction are as follows:

- The forex symbol is defined, in ISO codes, as "held-currency.new-currency" (e.g. "USD.GBP" indicates a desire to convert a held amount of USD to GPB)
- Side is defined in terms of whether the OrderQty is currency required or currency held (e.g. a forex order can be expressed as either "S 3,200,000 USD for GPB" or "B 2,000,000 GBP for USD", at a rate of 1.6 USD/GBP, these trades are equivalent in that the broker is receiving 3,200,000 USD and delivering 2,000,000 GBP)
- OrdType = Forex

The format for the new order message is as follows:

New Order - Single

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = D
11	ClOrdID	Y	
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
1	Account	N	
63	SettlmntTyp	N	Absence of this field is interpreted as Regular.
64	FutSettDate	N	Required when SettlmntTyp = 6 (Future) or SettlmntTyp = 8 (Sellers Option)
21	HandlInst	Y	
18	ExecInst	N	Can contain multiple instructions, space delimited.
110	MinQty	N	
111	MaxFloor	N	
100	ExDestination	N	
81	ProcessCode	N	Used to identify soft trades at order entry.
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
140	PrevClosePx	N	Useful for verifying security identification
54	Side	Y	
114	LocateReqd	N	Required for short sell orders

38	OrderQty	Y	
40	OrdType	Y	
44	Price	N	Required for limit OrdTypes
99	StopPx	N	Required for stop OrdTypes
15	Currency	N	Message without currency field is interpreted as US dollars
23	IOIid	N	Required for Previously Indicated Orders (OrdType=E)
117	QuoteID	N	Required for Previously Quoted Orders (OrdType=D)
59	TimeInForce	N	Absence of this field indicates Day order
126	ExpireTime	N	Required if TimeInForce = GTD
12	Commission	N	
13	CommType	N	
47	Rule80A	N	
121	ForexReq	N	Indicates that broker is requested to execute a Forex accommodation trade in conjunction with the security trade.
120	SettlCurrency	N	Required if ForexReq = Y.
58	Text	N	
	Standard Trailer	Y	

Execution Reports -

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 as orders are worked
5. reject orders
6. report miscellaneous fees calculations associated with a trade

NOTE: Execution reports do not replace the end-of-day confirm. Execution reports are to be regarded only as replacements for the existing fill messages currently communicated via telephone.

Each execution message will contain information that will describe the current state of the order and execution status as understood by the broker.

Execution report messages can be transmitted as transaction types (ExecTransType) NEW, CANCEL, CORRECT or STATUS. Transaction types CANCEL and CORRECT modify the state of the message identified in field ExecRefID. Transaction type STATUS indicates that the execution message contains no new information, only summary information regarding order status.

- The NEW transaction type indicates that this message represents a new order, a change in status of the order, or a new fill against an existing order. The combination of the ExecTransType and OrdStatus fields will indicate how the message is to be applied to an order.
- The CANCEL transaction type applies at the execution level. The Cancel transaction will be used to cancel an execution which has been reported in error. The canceled execution will be identified in the ExecRefID field.
- The CORRECT transaction type applies at the execution level and is used to modify an incorrectly reported fill. The incorrect execution will be identified in the ExecRefID field. *Note: Data reported in the CumQty and AvgPx fields represent the status of the order as of the time of the correction, not as of the time of the originally reported execution.*

The OrdStatus field is used to identify the status of the current order. The order statuses are as follows:

New	Outstanding order with no executions
Partially Filled	Outstanding order with executions and remaining quantity
Filled	Order completely filled, no remaining quantity
Done for Day	Order not, or partially, filled; no further executions forthcoming
Canceled	Canceled order with or without executions
Replaced	Replaced order with or without executions
Pending Cancel/Replace	Order with cancel request pending, used to confirm receipt of cancel or replace request. DOES NOT INDICATE THAT THE ORDER HAS BEEN CANCELED OR REPLACED.
Stopped	Order has been stopped at the exchange
Rejected	Order has been rejected by broker. NOTE: An order can be rejected subsequent to order acknowledgment, i.e. an order can pass from New to Rejected status.

Suspended	Order has been placed in suspended state at the request of the client.
Pending New	Order has been received by brokers system but not yet accepted for execution. An execution message with this status will only be sent in response to a Status Request message.
Expired	Order has been canceled in broker's system due to time in force instructions.
Calculated	Order has been completed for the day (either filled or done for day). Miscellaneous fees have been calculated and reported in this execution message

NOTE: The canceled and replaced order status is set in response to accepted cancel and replace requests. These requests are only acted upon when there is an outstanding order quantity. Requests to replace OrderQty to a level less than the CumQty will be rejected (OrderQty = CumQty + LeavesQty). Requests to change price on a filled order will be rejected (see Order Cancel Reject message type).

The CumQty and AvgPx fields should be calculated to reflect the fills on all versions of an order. For example, if partially filled order A were replaced by order B, the CumQty and AvgPx on order B's fills should represent the fills on order A plus those on order B.

The field ClOrdID is provided for institutions to affix an identification number to an order to coincide with internal systems. The OrderId field is populated with the broker-generated order number.

The execution report message format is as follows:

Execution Report

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 8
37	OrderID	Y	
11	ClOrdID	N	Required for executions against electronically submitted orders which were assigned an ID by the institution. Not required for orders manually entered by the broker.
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
66	ListID	N	Required for executions against orders which were submitted as part of a list.
17	ExecID	Y	
20	ExecTransType	Y	
19	ExecRefID	N	Required for Cancel and Correct ExecTransType messages
39	OrdStatus	Y	
103	OrdRejReason	N	For optional use with OrdStatus = 8 (Rejected)
1	Account	N	Required for executions against electronically submitted orders *which were assigned an account by the institution*
63	SettlmntTyp	N	Absence of this field is interpreted as Regular.

64	FutSettDate	N	Required when SettlmntTyp = 6 (Future) or SettlmntTyp = 8 (Sellers Option)
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
54	Side	Y	
38	OrderQty	Y	
40	OrdType	N	
44	Price	N	
99	StopPx	N	Required for OrdType = 4 (Stop Limit).
15	Currency	N	Message without currency field is interpreted as US dollars
59	TimeInForce	N	Absence of this field indicates Day order
126	ExpireTime	N	Required if TimeInForce = GTD
18	ExecInst	N	Can contain multiple instructions, space delimited.
47	Rule80A	N	
32	LastShares	Y	Not required ExecTransType = 3 (Status)
31	LastPx	Y	Not required for ExecTransType = 3 (Status)
30	LastMkt	N	
29	LastCapacity	N	
14	CumQty	Y	
6	AvgPx	Y	
75	TradeDate	N	Used when reporting other than current day trades.
60	TransactTime	N	
113	ReportToExch	N	
12	Commission	N	
13	CommType	N	
136	NoMiscFees	N	Required if any miscellaneous fees are reported. Indicates number of repeating entries.
137	***MiscFeeAmt***	***N***	Required if NoMiscFees > 0
138	***MiscFeeCurr***	***N***	Required if NoMiscFees > 0
139	***MiscFeeType***	***N***	Required if NoMiscFees > 0
118	NetMoney	N	Required if miscellaneous fees are reported, in currency of execution
119	SettlCurrAmt	N	Used to report results of forex accommodation trade

120	SettlCurrency	N	Used to report results of forex accommodation trade
58	Text	N	
	Standard Trailer	Y	

Don't Know Trade (DK) -

The Don't Know Trade (DK) message is used to notify a trading partner that an electronically received execution has been rejected. This message can be thought of as an execution reject message.

This message has special utility when dealing with one-way execution reporting, if the initial Order Acknowledgment message (LastShares=0 and OrdStatus=New) does not match an existing order this message can be used to notify the broker of a potential problem order.

Note that the decision to DK an execution lies with the institution; some of the mismatches listed in the DKReason field may be acceptable and will not require a DK messages to be generated.

The Don't Know Trade (DK) format is as follows:

Don't Know Trade (DK)

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = Q
37	OrderID	N	Broker Order Id as identified on problem execution
17	ExecID	N	Execution Id of problem execution
127	DKReason	Y	
55	Symbol	Y	
54	Side	Y	
38	OrderQty	Y	
32	LastShares	Y	
31	LastPx	Y	
58	Text	N	
	Standard Trailer	Y	

Order Cancel/Replace Request -

The order cancel/replace request is used to change the parameters of an existing order.

Do not use this message to cancel the remaining quantity of an outstanding order, use the Cancel Request message for this purpose.

It is recommended that the Cancel/Replace Request message be used to partially cancel (reduce) an order.

The request will only be accepted if the order can successfully be pulled back from the exchange floor without executing. Cancel/Replace requests which cannot be processed will be rejected using the

Cancel Reject message; the ClOrdId of the *replacement* order is inserted in the ClOrdId field of the Cancel Reject message for identification.

Only a limited number of fields can be changed via the cancel/replace request message. All other fields should be retransmitted as sent in the original order. These fields are:

- ExecInst
- OrderQty
- OrdType
- Price
- HandlInst
- TimeInForce
- ExpireTime
- MinQty
- MaxFloor

When modifying ExecInst fields in a replacement order, it is necessary to redeclare all ExecInst in the replacement order. ExecInst's will not be carried forward from the original order to the replacement unless redeclared.

The format of the cancel request message is:

Cancel/Replace Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = G
37	OrderID	N	Unique identifier of *original* order as assigned by broker
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
41	OrigClOrdID	Y	Unique identifier of *original* order as assigned by institution.
11	ClOrdID	Y	Unique identifier of *replacement* order as assigned by institution. Note that this identifier will be used in ClOrdID field of the Cancel Reject Message if the replacement request is rejected.
66	ListID	N	Required for List Orders
1	Account	N	
63	SettlmntTyp	N	Absence of this field is interpreted as Regular.
64	FutSettDate	N	Required when SettlmntTyp = 6 (Future) or SettlmntTyp = 8 (Sellers Option)
21	HandlInst	Y	Must match original order
18	ExecInst	N	Can contain multiple instructions, space delimited. Replacement order must be created with new parameters (i.e. original order values will not be brought forward to replacement order unless redefined within this message).
110	MinQty	N	
111	MaxFloor	N	
100	ExDestination	N	
55	Symbol	Y	Must match original order
65	SymbolSfx	N	
48	SecurityID	N	Must match original order
22	IDSource	N	Must match original order
106	Issuer	N	
107	SecurityDesc	N	
54	Side	Y	Must match original side, however, Buy and Buy Minus can be interchanged as well as Sell and Sell Plus
38	OrderQty	Y	
40	OrdType	Y	
44	Price	N	Required for limit OrdTypes
99	StopPx	N	Required for stop OrdTypes
15	Currency	N	Message without currency field is interpreted as US dollars. Must match original order.

59	TimeInForce	N	Absence of this field indicates Day order
126	ExpireTime	N	Required if TimeInForce = GTD
12	Commission	N	
13	CommType	N	
47	Rule80A	N	Must match original order
121	ForexReq	N	Indicates that broker is requested to execute a Forex accommodation trade in conjunction with the security trade.
120	SettlCurrency	N	Required if ForexReq = Y.
58	Text	N	
	Standard Trailer	Y	

Order Cancel Request -

The order cancel request message is used to request the cancellation of all or part of the remaining quantity of an existing order. The CxlType field is used to distinguish if all or part of the outstanding quantity is to be canceled.

Although the Order Cancel Request message can be used to partially cancel (reduce) an order it is recommended that the Cancel/Replace Request message be used for that purpose.

The request will only be accepted if the order can successfully be pulled back from the exchange floor without executing.

Note that a cancel request is assigned an order id and is treated as a separate entity. If rejected, the order id of the cancel request will be sent in the Cancel Reject message. The OrderID assigned to the cancel request must be unique amongst the OrderID's assigned to regular orders and replacement orders.

The format of the cancel request message is:

Order Cancel Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = F
41	OrigClOrdID	Y	Unique ID of original order as assigned by institution
37	OrderID	N	Broker ID of original order
11	ClOrdID	Y	Unique ID of cancel request as assigned by the institution.
66	ListID	N	Required for List Orders
125	CxlType	Y	
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
55	Symbol	Y	
65	SymbolSfx	N	

48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
54	Side	Y	
38	OrderQty	Y	Original OrderQty for CxlType=F or new OrderQty for CxlType=P.
58	Text	N	
	Standard Trailer	Y	

Order Cancel Reject -

The order cancel reject message is issued by the broker upon receipt of a cancel request or cancel/replace request message which cannot be honored. Requests to change price or decrease quantity are executed only when an outstanding quantity exists; orders which are filled cannot be changed.

When rejecting a Cancel/Replace Request, the ClOrdID of the *replacement* order in the request message is inserted in the ClOrdID field of the Cancel Reject message for identification.

The execution message will be used to respond to accepted cancel request and cancel/replace request messages.

The order cancel reject message format is as follows:

Order Cancel Reject

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 9
37	OrderID	Y	
11	ClOrdID	Y	Unique order id assigned by institution to the cancel request or to the *replacement* order.
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
66	ListID	N	Required for rejects against orders which were submitted as part of a list.
102	CxlRejReason	N	
58	Text	N	
	Standard Trailer	Y	

Order Status Request -

The order status request message is used by the institution to generate an order status message back from the broker.

The format of the order status request message is:

Order Status Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = H
37	OrderID	N	
11	ClOrdID	Y	
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
55	Symbol	Y	
65	SymbolSfx	N	
106	Issuer	N	
107	SecurityDesc	N	
54	Side	Y	
	Standard Trailer	Y	

Allocation -

The allocation record is used to instruct a broker on how to allocate executed shares to sub-accounts. The allocation record can also be used as a confirmation message through which third parties can communicate execution and settlement instructions between trading partners.

An allocation message can be submitted as new, cancel or replace. The AllocTransType field indicates the purpose of the message. When submitting replace or cancel AllocTransType messages the RefAllocID field is required. Replacement allocation messages must contain all data for the replacement allocation.

The allocation record contains repeating fields for each order, sub-account and individual execution; the repeating fields are shown below in typeface ***Bold-Italic.*** The relative position of the repeating fields is important in this record, i.e. each instance of allocation must be in the order shown below.

- The total shares allocated must equal the Shares value which must equal the total executed quantity of the original order. If present, the total shares in the execution section must also be equal to this value.
- The number of sub-account instances is indicated in NoAllocs.
- Multiple orders can be combined for allocation by identifying the number of orders in the NoOrders field and each individual order in the OrderID fields. Combined orders must have the same ticker, trade date, settlement date and side.

Allocation

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = J
70	AllocID	Y	
71	AllocTransTyp	Y	
72	RefAllocID	N	Required for AllocTransType = R (Replace) or C (Cancel)
73	NoOrders	Y	Indicates number of orders to be combined for allocation. If order(s) were manually delivered set to 1 (one).
11	***ClOrdID***	Y	Order ID assigned by client if order(s) were electronically delivered and executed. If order(s) were manually delivered this field should contain string "MANUAL".
37	***OrderID***	N	
66	***ListID***	N	Required for List Orders.
105	***WaveNo***	N	
124	NoExecs	N	Indicates number of individual execution record groups to follow. Absence of this field indicates that no individual execution records are included.
17	***ExecID***	N	
32	***LastShares***	N	Number of shares in individual execution. Required if NoExecs > 0
31	***LastPx***	N	Price of individual execution. Required if NoExecs > 0
30	***LastMkt***	N	Market of individual execution.

54	Side	Y	
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
53	Shares	Y	Total number of shares allocated to all accounts
6	AvgPx	Y	
15	Currency	N	Currency of AvgPx, absence of this field indicates US dollars
74	AvgPrxPrecision	N	Absence of this field indicates that default precision arranged by the broker/institution is to be used
75	TradeDate	Y	
60	TransactTime	N	Date/time when allocation is generated
63	SettlmntTyp	N	Absence of this field is interpreted as Regular
64	FutSettDate	N	Required with SettlmntTyp other than regular
118	NetMoney	N	Expressed in same currency as AvgPx
136	NoMiscFees	N	Required if any miscellaneous fees are reported. Indicates number of repeating entries
137	***MiscFeeAmt***	N	Required if NoMiscFees > 0
138	***MiscFeeCurr***	N	Required if NoMiscFees > 0
139	***MiscFeeType***	N	Required if NoMiscFees > 0
119	SettlCurrAmount	N	
120	SettlCurrency	N	
77	OpenClose	N	
58	Text	N	
78	NoAllocs	Y	Indicates number of allocation groups to follow.
79	***AllocAccount***	Y	
80	***AllocShares***	Y	
81	***ProcessCode***	N	
76	***ExecBroker***	N	Required for step-in and step-out trades
109	***ClientID***	N	Used for firm identification in third-party transactions.
12	***Commission***	N	
13	***CommType***	N	
85	***NoDlvyInst***	N	Repeating group within allocation group
92	***BrokerOfCredit***	N	

86	***DlvyInst***	N	Required if NoDlvyInst > 0
	Standard Trailer	Y	

Allocation ACK -

The allocation ACK record is used by the broker to acknowledge the receipt and status of an allocation record received from the institution.

It is possible that multiple Allocation ACK messages can be generated for a single allocation to detail the receipt and then the acceptance or rejection of the allocation.

Allocation ACK

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = P
109	ClientID	N	Used for firm identification in third-party transactions.
76	ExecBroker	N	Used for firm identification in third-party transactions.
70	AllocID	Y	
75	TradeDate	Y	
60	TransactTime	N	Date/Time allocation Ack generated
87	AllocStatus	Y	
88	AllocRejCode	N	Required for AllocStatus = 1 (rejected)
58	Text	N	Can include explanation for AllocRejCode = 7 (other)
	Standard Trailer	Y	

New Order List -

The new order list message type is used by institutions wishing to electronically submit lists of related orders to a broker for execution.

The New Order List is intended for use in *staging* lists to be executed by the broker. If the institution wishes to work a list using the broker's execution services the orders should be submitted as individual New Order - Single's.

After staging, the list can be operated on in the following ways:

Execute: The broker can be instructed to release the list for execution by sending the List-Execute message.

Cancel: After the list has been staged with the broker, it can be canceled via the submission of the List Cancel message. If the list has not yet been submitted for execution, the List Cancel message will instruct the broker not to execute it, if the list is being executed, the List Cancel message should trigger the broker's system to generate cancel requests for the remaining quantities of each order within the list. Individual orders within the list can be canceled via the Order Cancel Request message.

Status: A status of the list can be requested via the submission of the List-Status Request message. The broker will respond with one or more List-Status messages which will report executed quantity, canceled quantity and average price for each order in the list.

Replace: Individual orders within the list can be replaced via Order Cancel/Replace Request messages.

Executions against orders within the list will *not* normally be reported as they occur. (If this feature is desired the institution and broker should arrange for this reporting as a custom feature using the Execution message.) Executions against the list will be reported within the List-Status message.

The format for the new order list message is as follows:

New Order - List

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = E
66	ListID	Y	Must be unique, by customer, for the day
105	WaveNo	N	
67	ListSeqNo	Y	
68	ListNoOrds	Y	
69	ListExecInst	N	Include only in ListSeqNo = 1 message
11	ClOrdID	Y	
109	ClientID	N	Used for third-party transactions
76	ExecBroker	N	Used for third-party transactions
1	Account	N	
63	SettlmntTyp	N	Absence of this field is interpreted as Regular.

64	FutSettDate	N	Required when SettlmntTyp = 6 (Future) or SettlmntTyp = 8 (Sellers Option)
21	HandlInst	Y	
18	ExecInst	N	Can contain multiple instructions, space delimited.
110	MinQty	N	
111	MaxFloor	N	
100	ExDestination	N	
81	ProcessCode	N	
55	Symbol	Y	
65	SymbolSfx	N	
48	SecurityID	N	
22	IDSource	N	
106	Issuer	N	
107	SecurityDesc	N	
140	PrevClosePx	N	Useful for verifying security identification
54	Side	Y	
114	LocateReqd	N	Required for short sell orders
38	OrderQty	Y	
40	OrdType	Y	
44	Price	N	Required for limit OrdTypes
99	StopPx	N	Required for stop OrdTypes
15	Currency	N	Message without currency field is interpreted as US dollars
59	TimeInForce	N	Absence of this field indicates Day order
126	ExpireTime	N	Required in TimeInForce = GTD
12	Commission	N	
13	CommType	N	
47	Rule80A	N	
121	ForexReq	N	Indicates that broker is requested to execute a Forex accommodation trade in conjunction with the security trade.
120	SettlCurrency	N	Required if ForexReq = Y.
58	Text	N	
	Standard Trailer	Y	

List Status -

The list status message is issued as the response to a List Status Request message and indicates the current state of the orders within the list as they exists at the broker's site.

Orders within the list are statused at the summary level. Individual executions are not reported, rather, the current state of the order is reported.

The message contains repeating fields for each order; the repeating fields are shown below in typeface ***Bold-Italic.*** The relative position of the repeating fields is important in this record, i.e. each instance of ClOrdID, CumQty, CxlQty and AvgPx must be in the order shown below.

Each list status message will report on only a maximum of 50 orders; if the list contains more than 50 orders multiple status messages will be required.

The list status message format is as follows:

List Status

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = N
66	ListID	Y	
105	WaveNo	N	
82	NoRpts	Y	Total number of messages required to status complete list.
83	RptSeq	Y	Sequence number of this report message.
73	NoOrders	Y	Number of orders statused in this message, i.e. number of repeating groups to follow.
11	***ClOrdID***	***Y***	
14	***CumQty***	***Y***	
84	***CxlQty***	***Y***	
6	***AvgPx***	***Y***	
	Standard Trailer	Y	

List Execute -

The list execute message type is used by institutions to instruct the broker to begin execution of a previously submitted list.

The format for the list execute message is as follows:

List Execute

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = L
66	ListID	Y	Must be unique, by customer, for the day
105	WaveNo	N	
58	Text	N	
	Standard Trailer	Y	

List Cancel Request -

The list cancel request message type is used by institutions wishing to cancel previously submitted lists either before or during execution.

After the list has been staged with the broker, it can be canceled via the submission of the List Cancel message. If the list has not yet been submitted for execution, the List Cancel message will instruct the broker not to execute it, if the list is being executed, the List Cancel message should trigger the broker's system to generate cancel requests for the remaining quantities of each order within the list. Individual orders within the list can be canceled via the Order Cancel Request message.

The format for the list - cancel request message is as follows:

List Cancel Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = K
66	ListID	Y	
105	WaveNo	N	
58	Text	N	
	Standard Trailer	Y	

List Status Request -

The list status request message type is used by institutions to instruct the broker to generate status messages for a list.

The format for the list - status request message is as follows:

List Status Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = M
66	ListID	Y	
105	WaveNo	N	
58	Text	N	
	Standard Trailer	Y	

Field Definitions

The following is a catalog of fields used to define the application and session protocol messages.

Field ID (TAG)	Field Name	Format	Description
1	Account	char	Account mnemonic as agreed between broker and institution.
2	AdvId	int	Unique identifier of advertisement message
3	AdvRefID	int	Reference identifier used with CANCEL and REPLACE transaction types.
4	AdvSide	char	Broker's side of advertised trade Valid values: B = Buy S = Sell X = Cross T = Trade
5	AdvTransType	char	Identifies advertisement message transaction type Valid values: N = New C = Cancel R = Replace
6	AvgPx	float	Calculated average price of all fills on this order.
7	BeginSeqNo	int	Message sequence number of first record in range to be resent
8	BeginString	char	Identifies beginning of new message and protocol version. ALWAYS FIRST FIELD IN MESSAGE. (*Always unencrypted*) Valid values: FIX.4.0
9	BodyLength	int	Message length, in bytes, forward to the CheckSum field. ALWAYS SECOND FIELD IN MESSAGE. (*Always unencrypted*) Valid values: 0 - 9999

10	CheckSum	char	Three byte, simple checksum (see Appendix B for description). ALWAYS LAST FIELD IN RECORD; i.e. serves, with the trailing <SOH>, as the end-of-record delimiter. Always defined as three characters. (*Always unencrypted*)
11	ClOrdID	char	Unique identifier for Order as assigned by institution. Uniqueness must be guaranteed within a single trading day. Firms which electronically submit multi-day orders should consider embedding a date within the ClOrderID field to assure uniqueness across days.
12	Commission	float	Commission
13	CommType	char	Commission type Valid values: 1 = per share 2 = percentage 3 = absolute
14	CumQty	int	Total number of shares filled. Valid values: (0 - 1000000000)
15	Currency	char	Identifies currency used for price, Absence of this field in a message is interpreted as US dollars. See Appendix A for information on obtaining valid values.
16	EndSeqNo	int	Message sequence number of last record in range to be resent. If request is for a single record BeginSeqNo = EndSeqNo. If request is for all messages subsequent to a particular message, EndSeqNo = "999999"
17	ExecID	int	Unique identifier of execution message as assigned by broker (will be 0 (zero) for ExecTransType=3 (Status)). Uniqueness must be guaranteed within a single trading day. Firms which accept multi-day orders should consider embedding a date within the ExecID field to assure uniqueness across days.

18	ExecInst	char	Instructions for order handling on exchange trading floor. If more than one instruction is applicable to an order, this field can contain multiple instructions separated by space. Valid values: 1 = Not held 2 = Work 3 = Go along 4 = Over the day 5 = Held 6 = Participate don't initiate 7 = Strict scale 8 = Try to scale 9 = Stay on bidside 0 = Stay on offerside A = No cross B = OK to cross C = Call first D = Percent of volume E = Do not increase - DNI F = Do not reduce - DNR G = All or none - AON I = Institutions only L = Last peg (last sale) M = Mid-price peg (midprice of inside quote) N = Non-negotiable O = Opening peg P = Market peg R = Primary peg (primary market - buy at bid/sell at offer) S = Suspend
19	ExecRefID	int	Reference identifier used with Cancel and Correct transaction types.
20	ExecTransType	char	Identifies transaction type Valid values: 0 = New 1 = Cancel 2 = Correct 3 = Status
21	HandlInst	char	Instructions for order handling on Broker trading floor Valid values: 1 = Automated execution order, private, no Broker intervention 2 = Automated execution order, public, Broker intervention OK 3 = Manual order, best execution

22	IDSource	char	Identifies class of alternative SecurityID Valid values: 1 = CUSIP 2 = SEDOL 3 = QUIK 4 = ISIN number 5 = RIC code 100+ are reserved for private security identifications
23	IOIid	int	Unique identifier of IOI message.
24	IOIOthSvc	char	Indicates if, and on which other services, the indication has been advertised. Each character represents an additional service (e.g. if on Bridge and Autex, field = BA, if only on Autex, field = A) Valid values: A = Autex B = Bridge
25	IOIQltyInd	char	Relative quality of indication Valid values: L = Low M = Medium H = High
26	IOIRefID	int	Reference identifier used with CANCEL and REPLACE, transaction types.
27	IOIShares	char	Number of shares in numeric or relative size. Valid values: 0 - 1000000000 S = Small M = Medium L = Large
28	IOITransType	char	Identifies IOI message transaction type Valid values: N = New C = Cancel R = Replace

29	LastCapacity	char	Broker capacity in order execution Valid values: 1 = Agent 2 = Cross as agent 3 = Cross as principal 4 = Principal
30	LastMkt	char	Market of execution for last fill Valid values: See Appendix C
31	LastPx	float	Price of last fill. Field not required for ExecTransType = 3 (Status)
32	LastShares	int	Quantity of shares bought/sold on this fill. Field not required for ExecTransType = 3 (Status)
33	LinesOfText	int	Identifies number of lines of text body
34	MsgSeqNum	int	Integer message sequence number. Valid values: 0 - 999999

35	MsgType	char	Defines message type. ALWAYS THIRD FIELD IN MESSAGE. (*Always unencrypted*) *Note: A "U" as the first character in the MsgType field indicates that the message format is privately defined between the sender and receiver.* Valid values: 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 F = Order Cancel Request G= Order Cancel/Replace Request H= Order Status Request J = Allocation K = List Cancel Request L = List Execute M = List Status Request N = List Status P = Allocation ACK Q = Don't Know Trade (DK) R = Quote Request S = Quote
36	NewSeqNo	int	New sequence number Valid values: 0 - 999999
37	OrderID	char	Unique identifier for Order as assigned by broker. Uniqueness must be guaranteed within a single trading day. Firms which accept multi-day orders should consider embedding a date within the OrderID field to assure uniqueness across days.
38	OrderQty	int	Number of shares ordered Valid values: (0 - 1000000000)

39	OrdStatus	char	Identifies current status of order. Valid values: 0 = New 1 = Partially filled 2 = Filled 3 = Done for day 4 = Canceled 5 = Replaced 6 = Pending Cancel/Replace 7 = Stopped 8 = Rejected 9 = Suspended A = Pending New B = Calculated C = Expired
40	OrdType	char	Order type. Valid values: 1 = Market 2 = Limit 3 = Stop 4 = Stop limit 5 = Market on close 6 = With or without 7 = Limit or better 8 = Limit with or without 9 = On basis A = On close B = Limit on close C = Forex D = Previously quoted E = Previously indicated P = Pegged (requires ExecInst = L, R, M, P or O)
41	OrigClOrdID	char	Original order id as assigned by the institution, used to identify original order in cancel and cancel/replace requests.
42	OrigTime	time	Time of message origination (always expressed in GMT)
43	PossDupFlag	char	Indicates possible retransmission of message with this sequence number Valid values: Y = Possible duplicate N = Original transmission
44	Price	float	Price per share Valid values: 0 - 99999999.9999

45	RefSeqNum	int	Reference message sequence number Valid values: 0 - 999999
46	RelatdSym	char	Symbol of issue related to story. Can be repeated within message to identify multiple companies.
47	Rule80A	char	Indicates order type upon which exchange Rule 80A is applied. Valid values: A = Agency single order I = Individual Investor, single order D = Program Order, index arb, for Member firm/org C = Program Order, non-index arb, for Member firm/org J = Program Order, index arb, for individual customer K = Program Order, non-index arb, for individual customer U = Program Order, index arb, for other agency Y = Program Order, non-index arb, for other agency M = Program Order, index arb, for other member N = Program Order, non-index arb, for other member W = All other orders as agent for other member
48	SecurityID	char	CUSIP or other alternate security identifier
49	SenderCompID	char	Assigned value used to identify firm sending message.
50	SenderSubID	char	Assigned value used to identify specific message originator (desk, trader, etc.)
51	SendingDate *(not used)*	date	Field not presently used. Included here as reference to previous versions.
52	SendingTime	time	Time of message transmission (always expressed in GMT)
53	Shares	int	Number of shares Valid values: 0 - 1000000000
54	Side	char	Side of order Valid values: 1 = Buy 2 = Sell 3 = Buy minus 4 = Sell plus 5 = Sell short 6 = Sell short exempt

55	Symbol	char	Ticker symbol
56	TargetCompID	char	Assigned value used to identify receiving firm.
57	TargetSubID	char	Assigned value used to identify specific individual or unit intended to receive message. "ADMIN" reserved for administrative messages not intended for a specific user.
58	Text	char	Free format text string
59	TimeInForce	char	Specifies how long the order remains in effect. Absence of this field is interpreted as DAY. Valid values: 0 = Day 1 = Good Till Cancel (GTC) 2 = At the Opening (OPG) 3 = Immediate or Cancel (OC) 4 = Fill or Kill (FOK) 5 = Good Till Crossing (GTX) 6 = Good Till Date
60	TransactTime	time	Time of execution/order creation (expressed in GMT)
61	Urgency	char	Urgency flag Valid values: 0 = Normal 1 = Flash 2 = Background
62	ValidUntilTime	time	Indicates expiration time of indication message (always expressed in GMT)
63	SettlmntTyp	char	Indicates order settlement period. Absence of this field is interpreted as Regular. Regular is defined as the default settlement period for the particular security on the exchange of execution. Valid values: 0 = Regular 1 = Cash 2 = Next Day 3 = T+2 4 = T+3 5 = T+4 6 = Future 7 = When Issued 8 = Sellers Option 9 = T+ 5

64	FutSettDate	date	Specific date of trade settlement in YYYYMMDD format. Required when SettlmntTyp = 6 (Future) or SettlmntTyp = 8 (Sellers Option). (expressed in local time at place of settlement)
65	SymbolSfx	char	Additional information about the security (e.g. preferred, warrants, etc.). Absence of this field indicates common. Valid values: As defined in the NYSE Stock and bond Symbol Directory and in the AMEX Fitch Directory
66	ListID	char	Customer assigned listUnique identifier for list as assigned by institution, used to associate multiple individual orders. Uniqueness must be guaranteed within a single trading day. Firms which generate multi-day orders should consider embedding a date within the ListID field to assure uniqueness across days.
67	ListSeqNo	int	Sequence of individual order within list (i.e. *ListSeqNo* of *ListNoOrds,* 2 of 25, 3 of 25, . . .)
68	ListNoOrds	int	Total number of orders within list (i.e. *ListSeqNo* of *ListNoOrds,* e.g. 2 of 25, 3 of 25, . . .)
69	ListExecInst	char	Free format text message containing list handling and execution instructions.
70	AllocID	int	Unique identifier for allocation record.
71	AllocTransType	char	Identifies allocation transaction type Valid values: 0 = New 1 = Replace 2 = Cancel
72	RefAllocID	int	Reference identifier to be used with Replace and Cancel AllocTransType records.
73	NoOrders	int	Indicates number of orders to be combined for average pricing and allocation.
74	AvgPrxPrecision	int	Indicates number of decimal places to be used for average pricing. Absence of this field indicates that default precision arranged by the broker/institution is to be used.

75	TradeDate	date	Indicates date of trade referenced in this record in YYYYMMDD format. Absence of this field indicates current day (expressed in local time at place of trade).
76	ExecBroker	char	Identifies executing / give-up broker. Standard NASD market-maker mnemonic is preferred.
77	OpenClose	char	For options only.
78	NoAllocs	int	Number of AllocAccount/AllocShares/ProcessCode instances included in allocation record.
79	AllocAccount	char	Sub-account mnemonic
80	AllocShares	int	Number of shares to be allocated to specific sub-account
81	ProcessCode	char	Processing code for sub-account. Absence of this field in AllocAccount / AllocShares / ProcessCode instance indicates regular trade. Valid values: 0 = regular 1 = soft dollar 2 = step-in 3 = step-out 4 = soft-dollar step-in 5 = soft-dollar step-out 6 = plan sponsor
82	NoRpts	int	Total number of reports within series.
83	RptSeq	int	Sequence number of message within report series.
84	CxlQty	int	Total number of shares canceled for this order.
85	NoDlvyInst	int	Number of delivery instruction fields to follow
86	DlvyInst	char	Free format text field to indicate delivery instructions

87	AllocStatus	int	Identifies status of allocation. Valid values: 0 = accepted (successfully processed) 1 = rejected 2 = partial accept 3 = received (received, not yet processed)
88	AllocRejCode	int	Identifies reason for rejection. Valid values: 0 = unknown account(s) 1 = incorrect quantity 2 = incorrect average price 3 = unknown executing broker mnemonic 4 = commission difference 5 = unknown OrderID 6 = unknown ListID 7 = other
89	Signature	data	Electronic signature
90	SecureDataLen	int	Length of encrypted message
91	SecureData	data	Actual encrypted data stream
92	BrokerOfCredit	char	Broker to receive trade credit
93	SignatureLength	int	Number of bytes in signature field.
94	EmailType	char	Email message type. Valid values: 0 = New 1 = Reply 2 = Admin Reply
95	RawDataLength	int	Number of bytes in raw data field.
96	RawData	data	Unformatted raw data, can include bitmaps, word processor documents, etc.
97	PossResend	char	Indicates that message may contain information that has been sent under another sequence number.

98	EncryptMethod	int	Method of encryption. Valid values: 0 = None / other 1 = PKCS (proprietary) 2 = DES (EBC mode) 3 = PKCS/DES (proprietary) 4 = PGP/DES (defunct) 5 = PGP/DES-MD5 (see app note on FIX home page) 6 = PEM/DES-MD5 (see app note on FIX home page)
99	StopPx	float	Price per share Valid values: 0 - 99999999.9999
100	ExDestination	char	Execution destination as defined by institution when order is entered. Valid values: See Appendix C *plus* 0 = none 4 = POSIT
102	CxlRejReason	int	Code to identify reason for cancel rejection. Valid values: 0 = Too late to cancel 1 = Unknown order
103	OrdRejReason	int	Code to identify reason for order rejection. Valid values: 0 = Broker option 1 = Unknown symbol 2 = Exchange closed 3 = Order exceeds limit 4 = Too late to enter

104	IOIQualifier	char	Code to qualify IOI use. Valid values: X = Crossing opportunity O = At the open M = More behind P = Taking a position V = Versus Q = Current quote C = At the close S = Portfolio show-n I = In touch with W = Indication - Working away A = All or none L = Limit T = Through the day
105	WaveNo	char	Identifier to aid in the management of multiple lists derived from a single, master list.
106	Issuer	char	Company name of security issuer (e.g. *International Business Machines)*
107	SecurityDesc	char	Security description.
108	HeartBtInt	int	Heartbeat interval (seconds)
109	ClientID	char	Firm identifier used in third party-transactions.
110	MinQty	int	Minimum quantity of an order to be executed.
111	MaxFloor	int	Maximum number of shares within an order to be shown on the exchange floor at any given time.
112	TestReqID	char	Identifier included in Test Request message to be returned in resulting Heartbeat
113	ReportToExch	char	Identifies party of trade responsible for exchange reporting. Valid values: Y = Indicates that party receiving message must report trade N = Indicates that party sending message will report trade

114	LocateReqd	char	Indicates whether the broker is to locate the stock in conjuction with a short sell order. Valid values: Y = Indicates the broker is responsible for locating the stock N = Indicates the broker is not required to locate
115	OnBehalfOfCompID	char	Assigned value used to identify firm originating message if the message was delivered by a third party i.e. the third party firm identifier would be delivered in the SenderCompID field and the firm originating the message in this field.
116	OnBehalfOfSubID	char	Assigned value used to identify specific message originator (desk, trader, etc.) if the message was delivered by a third party
117	QuoteID	char	Unique identifier for quote
118	NetMoney	float	Total amount due as the result of the transaction (e.g. for Buy order - principal + commission + fees) reported in currency of execution.
119	SettlCurrAmt	float	Total amount due expressed in settlement currency (includes the effect of the forex transaction)
120	SettlCurrency	char	Currency code of settlement denomination.
121	ForexReq	char	Indicates request for forex accommodation trade to be executed along with security transaction. Valid values: Y = Execute Forex after security trade N = Do not execute Forex after security trade
122	OrigSendingTime	time	Original time of message transmission (always expressed in GMT) when transmitting orders as the result of a resend request.
123	GapFillFlag	char	Indicates that the Sequence Reset message is replacing administrative or application messages which will not be resent. Valid values: Y = Gap Fill message, MsgSeqNum field valid N = Sequence Reset, ignore MsgSeqNum
124	NoExecs	int	No of execution record groups to follow.

125	CxlType	char	Defines if cancel is for part or all of the remaining quantity of an order. Valid values: P = partial cancel (reduce quantity) F = full remaining quantity
126	ExpireTime	time	Time/Date of order expiration (always expressed in GMT)
127	DKReason	char	Reason for execution rejection. Valid values: A = Unknown symbol B = Wrong side C = Quantity exceeds order D = No matching order E = Price exceeds limit Z = Other
128	DeliverToCompID	char	Assigned value used to identify the firm targeted to receive the message if the message is delivered by a third party i.e. the third party firm identifier would be delivered in the TargetCompID field and the ultimate receiver firm ID in this field.
129	DeliverToSubID	char	Assigned value used to identify specific message recipient (desk, trader, etc.) if the message is delivered by a third party
130	*IOINaturalFlag*	char	Indicates that IOI is the result of an existing agency order or a facilitation position resulting from an agency order, not from principal trading or order solicitation activity. Valid values: Y = Natural N = Not natural
131	QuoteReqID	char	Unique identifier for quote request
132	BidPx	float	Bid price/rate
133	OfferPx	float	Offer price/rate
134	BidSize	int	Quantity of bid
135	OfferSize	int	Quantity of offer

136	NoMiscFees	int	Number of repeating groups of miscellaneous fees
137	MiscFeeAmt	float	Miscellaneous fee value
138	MiscFeeCurr	char	Currency of miscellaneous fee
139	MiscFeeType	char	Indicates type of miscellaneous fee. Valid values: 1 = Regulatory (e.g. SEC) 2 = Tax 3 = Local Commission 4 = Exchange Fees 5 = Stamp 6 = Levy 7 = Other
140	PrevClosePx	float	Previous closing price of security.

FIX Field Index sorted by tag number:

1	Account	20	ExecTransType	39	OrdStatus	58	Text
2	AdvId	21	HandlInst	40	OrdType	59	TimeInForce
3	AdvRefID	22	IDSource	41	OrigClOrdID	60	TransactTime
4	AdvSide	23	IOIid	42	OrigTime	61	Urgency
5	AdvTransType	24	IOIOthSvc	43	PossDupFlag	62	ValidUntilTime
6	AvgPx	25	IOIQltyInd	44	Price	63	SettlmntTyp
7	BeginSeqNo	26	IOIRefID	45	RefSeqNum	64	FutSettDate
8	BeginString	27	IOIShares	46	RelatdSym	65	SymbolSfx
9	BodyLength	28	IOITransType	47	Rule80A	66	ListID
10	CheckSum	29	LastCapacity	48	SecurityID	67	ListSeqNo
11	ClOrdID	30	LastMkt	49	SenderCompID	68	ListNoOrds
12	Commission	31	LastPx	50	SenderSubID	69	ListExecInst
13	CommType	32	LastShares	51	SendingDate	70	AllocID
14	CumQty	33	LinesOfText	52	SendingTime	71	AllocTransType
15	Currency	34	MsgSeqNum	53	Shares	72	RefAllocID
16	EndSeqNo	35	MsgType	54	Side	73	NoOrders
17	ExecID	36	NewSeqNo	55	Symbol	74	AvgPrxPrecision
18	ExecInst	37	OrderID	56	TargetCompID	75	TradeDate
19	ExecRefID	38	OrderQty	57	TargetSubID	76	ExecBroker

77 OpenClose
78 NoAllocs
79 AllocAccount
80 AllocShares
81 ProcessCode
82 NoRpts
83 RptSeq
84 CxlQty
85 NoDlvyInst
86 DlvyInst
87 AllocStatus
88 AllocRejCode
89 Signature
90 SecureDataLen
91 SecureData
92 BrokerOfCredit
93 SignatureLength
94 EmailType
95 RawDataLength
96 RawData
97 PossResend
98 EncryptMethod
99 StopPx
100 ExDestination
102 CxlRejReason
103 OrdRejReason
104 IOIQualifier
105 WaveNo
106 Issuer
107 SecurityDesc
108 HeartBtInt
109 ClientID
110 MinQty
111 MaxFloor
112 TestReqID
113 ReportToExch
114 LocateReqd
115 OnBehalfOfID
116 OnBehalfOfSubID
117 QuoteID
118 NetMoney
119 SettlCurrAmt
120 SettlCurrency
121 ForexReq
122 OrigSendingTime
123 GapFillFlag
124 NoExecs
125 CxlType
126 ExpireTime
127 DKReason
128 DeliverToCompID
129 DeliverToSubID
130 IOINaturalFlag
131 QuoteReqID
132 BidPx
133 OfferPx
134 BidSize
135 OfferSize
136 NoMiscFees
137 MiscFeeAmt
138 MiscFeeCurr
139 MiscFeeType
140 PrevClosePx

FIX Field Index sorted by field name:

Appendix A

Valid Currency Codes

Currency codes used in FIX are those defined in ISO 4217 standard. To obtain the current valid list please contact the ISO 4217 secretariat at +44-181-996-9000.

Note: Prices defined in FIX messages should be made consistent with the currency code used. In some markets, prices are quoted as multiples or fractions of the currency, FIX messages should normalize the amount to coincide with the indicated code (e.g. UK securities are quoted in pence but must be represented in FIX messages as pounds).

Appendix B

CheckSum Calculation

The checksum of a FIX message is calculated by summing every byte of the message up to but not including the checksum field itself. This checksum is then transformed into a modulo 256 number for transmission and comparison. The checksum is calculated after all encryption is completed, i.e. the message as transmitted between parties is processed.

For transmission, the checksum must be sent as printable characters, so the checksum is transformed into three ASCII digits.

For example, if the checksum has been calculated to be 274 then the modulo 256 value is 22. This value would be transmitted a |10=022| where "10="is the tag for the checksum field.

A sample code fragment to generate the checksum field is as follows:

```
char *GenerateCheckSum( char *buf, long bufLen )
{
    static char tmpBuf[ 4 ];
    long idx;
    unsigned int cks;

    for( idx = 0L, cks = 0; idx < bufLen; cks += (unsigned int)buf[ idx++ ] );
    sprintf( tmpBuf, "%03d", (unsigned int)( cks % 256 ) );
    return( tmpBuf );
}
```

Appendix C

Reuters Exchange Mnemonics

Alberta Stock Exchange	AL
American Stock Exchange	A
Amman Stock Exchange	AM
Amsterdam Stock Exchange	AS
Australian Stock Exchange	AX
Bahrain Stock Exchange	BH
Basle Stock Exchange	BS
Barcelona Stock Exchange - Floor Trading	BC
Barcelona Stock Exchange - CATS Feed	MC
Belfox	b
Berlin Stock Exchange	BE
Berne Stock Exchange	BN
Bologna Stock Exchange	BL
Bombay Stock Exchange	BO
Bordeaux Stock Exchange	BD
Boston Stock Exchange	B
Bremen Stock Exchange	BM
Brussels Stock Exchange	BR
Chicago Board Options Exchange	W
Cincinnati Stock Exchange	C
Colombo Stock Exchange	CM
Copenhagen Stock Exchange	CO
Deutsche Terminboerse (DTB)	d
Dusseldorf Stock Exchange	D
European Options Exchange	E
Florence Stock Exchange	FL
Frankfurt Stock Exchange	F
Fukuoka Stock Exchange	FU
Geneva Stock Exchange	G
Genoa Stock Exchange	GE
Hamburg Stock Exchange	H
Hanover Stock Exchange	HA
Helsinki Stock Exchange	HE
Hiroshima Stock Exchange	HI
Hong Kong Stock Exchange	HK
Integrated Bourse Trading and Information System (IBIS)	IB
Istanbul Stock Exchange	IS
Jakarta Stock Exchange	JK
Japanese Securities Dealers Association	Q
Johannesburg Stock Exchange	J
Karachi Stock Exchange	KA
Korea Stock Exchange	KS
Kuala Lumpur Stock Exchange	KL
Kyoto Stock Exchange	KY
Lagos Stock Exchange	LG
Lausanne Stock Exchange	LA
Lille Stock Exchange	LI
London Stock Exchange	L
Luxembourg Stock Exchange	LU
Lyon Stock Exchange	LY
Madrid Stock Exchange - Floor Trading	MA
Madrid Stock Exchange - CATS Feed	MC
Marseille Stock Exchange	MS
MATIS	MT
MEFF Renta Variable	I
Mexican Stock Exchange	MX
Midwest Stock Exchange	MW
Milan Stock Exchange	MI
MONEP Paris Stock Options	p
Montreal Exchange	M
Munich Stock Exchange	MU
Muscat Stock Exchange	OM
Nancy Stock Exchange	NC
Nagoya Stock Exchange	NG
Nairobi Stock Exchange	NR
Nantes Stock Exchange	NT
Naples Stock Exchange	NA
NASDAQ	O
NASDAQ Dealers - International	OI
NASDAQ Dealers - Bulletin Board	OB
New York Stock Exchange	N
New Zealand Stock Exchange	NZ
Niigata Stock Exchange	NI
Osaka Stock Exchange	OS
Oslo Stock Exchange	OL
Pacific Stock Exchange	P
Palermo Stock Exchange	PL
Paris Stock Exchange	PA
Philadelphia Stock Exchange	PH
Philadelphia Stock Exchange Options	X
Rome Stock Exchange	RO
Sao Paulo Stock Exchange	SA
Sapporo Stock Exchange	SP
Singapore Stock Exchange	SI
Shanghai Stock Exchange	SS
Shenzhen Stock Exchange	SZ
Stockholm Options Market	o
Stockholm Stock Exchange	ST
Stuttgart Stock Exchange	SG
Swiss Options and Financial Futures Exchange (SOFFEX)	Z
Taiwan Stock Exchange	TW
Tel Aviv Stock Exchange	TA
Thailand Stock Exchange	BK
Third Market	TH
Tokyo Stock Exchange	T
Toronto Options Exchange	K
Toronto Stock Exchange	TO
Trieste Stock Exchange	TR
Tunis Stock Exchange	TN
Turin Stock Exchange	TU
Vancouver Stock Exchange	V
Venice Stock Exchange	VE
Vienna Stock Exchange	VI
Zimbabwe Stock Exchange	ZI
Zurich Stock Exchange	Z

Appendix D

Order State Change Matrices

The following matrices are included to clarify the sequence of messages and the status of orders involved in the submission and processing of new orders, executions, cancel requests and cancel/replace requests. These state diagrams are presented from the broker's view. (Note: x refers to the original order, y refers to the cancel/replacing order)

Filled Order:

Time	Message Received	Message Sent	OrdStatus X	OrdStatus Y
1	NewOrder (X)		Pending New (A)*	
2 (if rejected)		Execution	Rejected (8)	
2		Execution	New (0)	
3 (if partial fills, can be repeated many time)		Execution	Partial fill (1)	
4		Execution	Filled (2)	

Partially Filled Order:

Time	Message Received	Message Sent	OrdStatus X	OrdStatus Y
1	NewOrder (X)		Pending New (A)*	
2 (if rejected)		Execution	Rejected (8)	
2		Execution	New (0)	
3 (if partial fills, can be repeated many time)		Execution	Partial fill (1)	
4		Execution	Done for day (3)	

Canceled Order:

Time	Message Received	Message Sent	OrdStatus X	OrdStatus Y
1	NewOrder (X)		Pending New (A)*	
2 (if rejected)		Execution	Rejected (8)	
2		Execution	New (0)	
3	Cancel Request (Y)		New (0)*	Pending New (A)*
4 (if rejected)		Cancel Reject	New (0)*	Rejected (8)
4		Execution	Pending Cxl (6)*	New (0)
5 (if rejected)		Cancel Reject	New (0)*	Rejected (8)
5		Execution	Canceled (4)	Canceled (4)*

* information only transmitted as the result of an Order Status Request on this order

Replaced Order:

Time	Message Received	Message Sent	OrdStatus X	OrdStatus Y
1	NewOrder (X)		Pending New (A)*	
2 (if rejected)		Execution	Rejected (8)	

2		Execution	New (0)	
3	Cxl/Repl Request (Y)		New (0)*	Pending New (A)*
4 (if rejected by salesperson)		Cancel Reject	New (0)*	Rejected (8)
4		Execution	Pending Cxl (6)	Pending New (A)*
5 (if rejected by trader or exchange)		Cancel Reject	New (0)*	Rejected (8)
5		Execution	Replaced (4)*	New (0)

* information only transmitted as the result of an Order Status Request on this order