Newedge

FIX PROTOCOL
INDUSTRY-DRIVEN MESSAGING STANDARD℠
MEMBER

NEWEDGE FIX Protocol 4.3 – 4.4
Functional Specifications

Type	Fix Functional Specification
Reference	
Version	1.8
Date	26 May 2011
State	Official Document
Author	Satyaprakash Kurundkar
Review	Dominique Junker, Arnaud Prieur du Perray, Sylvain Maly

Distribution

Firm	People	Nb of copies	For Action	For info.

Modifications

Version	Date	Who	Object	Description
01.0	6 Sept 2007	G Colins	Creation	Creation
01.5	20 Apr 2009	S Maly	Update	Add new TOCOM order types
01.6	12 Jan 2010	S Maly	Update	Review of order types for Asian exchanges
01.7	15 Jan 2010	S Maly	Update	Remove Stop and Stop Limit orders from OSE and TSE
01.8	26 May 2010	Satyaprakash Kurundkar	Update	Valid values for tag 100 CME iLink Mandatory tags

Reference Documents

Reference	Version	Date	Title
Ullink FIX Specification	V3		

Conventions

In this document, the FIX messages are described as per the table below.

FIX messages conventions

Tag	Field Name	Req'd	Description
	Standard Header	**Y**	MsgType = A
X	**Required_Tag**	**Y**	'Y' in the *Required* column indicates that the tag is required.
Y	Conditionally_ Required_Tag	C	'C' in the *Required* column indicates that the tag is required only if certain conditions occur. Required conditions are explained in the *Description* column.
Z	Optional_Tag	N	'N' in the *Required* column indicates that the tag is not required, its absence will not cause a reject.
	Standard Trailer	**Y**	

Introduction

The purpose of this document is to give functional specifications of the FIX connectivity to NEWEDGE's FIX Engine.
The scope of the document has been reduced to only target the supported message types and contain by our FIX gateway (Supported fields and scope are sublimed).
Specific values expected by our system have 'Y' as value in the Req'd columns.
The FIX protocol specification can be found at http://www.fixprotocol.org.
Each time we describe a specific FIX Tag for a Client, we identify the Client by three letters XXX.

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. It should be noted that if an "unreliable" or non-stream protocol is used, the Logon, Logout, and ResendRequest message processing is particularly susceptible to unordered delivery and/or message loss.
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 with a field delimiter between fields in the stream. Tags are of data type *TagNum.*
All tags must have a value specified. Optional fields without values should simply not be specified in the FIX message. A Reject message is the appropriate response to a tag with no value.
Except where noted, fields within a message can be defined in any sequence (Relative position of a field within a message is inconsequential.) The exceptions to this rule are:

1. General message format is composed of the standard header followed by the body followed by the standard trailer.
2. The first three fields in the standard header are BeginString (tag #8) followed by BodyLength (tag #9) followed by MsgType (tag #35).

3. The last field in the standard trailer is the CheckSum (tag #10).
4. Fields within repeating data groups must be specified in the order that the fields are specified in the message definition within the FIX specification document. The NoXXX field where XXX is the field being counted specifies the number of repeating group instances that must immediately precede the repeating group contents.

In addition, certain fields of the data type *MultipleValueString* can contain multiple individual values separated by a space within the "value" portion of that field followed by a single "SOH" character (e.g. "18=2 9 C<SOH>" represents 3 individual values: '2', '9', and 'C').
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 considered more reliable. (A security warning should be generated).

Field Delimiter:
All fields (including those of data type *data i.e.* SecureData, RawData, SignatureData, XmlData, etc.) in a FIX message are terminated by a delimiter character. The non-printing, ASCII "SOH" (#001, hex: 0x01, referred to in this document as <SOH>), is used for field termination. Messages are delimited by the “SOH” character following the CheckSum field. All messages begin with the “8=FIX.x.y<SOH>” string and terminate with “10=nnn<SOH>“.
There shall be no embedded delimiter characters within fields except for data type *data*.

Repeating Groups:
It is permissible for fields to be repeated within a repeating group
(e.g. "384=2<SOH>372=6<SOH>385=R<SOH>372=7<SOH>385=R<SOH>" represents a repeating group with two repeating instances “delimited” by tag 372 (first field in the repeating group.).
If the repeating group is used, the first field of the repeating group is required. This allows implementations of the protocol to use the first field as a "delimiter" indicating a new repeating group entry.

- The NoXXX field which specifies the number of repeating group instances occurs once for a repeating group and must immediately precede the repeating group contents.
- If a repeating group field is listed as required, then it must appear in every repeated instance of that repeating group.
- Repeating groups are designated within the message definition via indentation and the symbol.

Some repeating groups are nested within another repeating group (potentially more than one level of nesting).

- Nested repeating groups are designated within the message definition via indentation and the symbol followed by another symbol.
- If a nested repeating group is used, then the outer repeating group must be specified

Data Types:

Data types (with the exception of those of type "data") 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"). Note that int values may contain leading zeros (e.g. “00023” = “23”).
  Examples:
  723 in field 21 would be mapped int as |21=723|.
  -723 in field 12 would be mapped int as |12=-723|

- **Length**: int field (see definition of “int” above) representing the length in bytes. Value must be positive.
- **NumInGroup:** int field (see definition of “int” above) representing the number of entries in a repeating group. Value must be positive.
- **SeqNum:** int field (see definition of “int” above) representing a message sequence number. Value must be positive.
- **TagNum:** int field (see definition of “int” above) representing a field's tag number when using FIX "Tag=Value" syntax. Value must be positive and may not contain leading zeros.
- **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. The number of decimal places used should be a factor of business/market needs and mutual agreement between counterparties. Note that float values may contain leading zeros (e.g. “00023.23” = “23.23”) and may contain or omit trailing zeros after the decimal point (e.g. “23.0” = “23.0000” = “23”).
- **Qty**: float field (see definition of “float” above) capable of storing either a whole number (no decimal places) of “shares” (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units).
- **Price**: float field (see definition of “float” above) representing a price. Note the number of decimal places may vary.
- **PriceOffset:** float field (see definition of “float” above) representing a price offset, which can be mathematically added to a "Price". Note the number of decimal places may vary and some fields such as LastForwardPoints may be negative.
- **Amt:** float field (see definition of “float” above) typically representing a Price times a Qty.
- **Percentage:** float field (see definition of “float” above) representing a percentage (e.g. .05 represents 5% and .9525 represents 95.25%). Note the number of decimal places may vary.
- **char:** Single character value, can include any alphanumeric character or punctuation except the delimiter. All char fields are case sensitive (i.e. m ≠ M).
- **Boolean:** a char field (see definition of “char” above) containing one of two values: ‘Y' = True/Yes
  'N' = False/No
- **String:** Alpha-numeric free format strings, can include any character or punctuation except the delimiter. All char fields are case sensitive (i.e. **morstatt ≠ Morstatt**).
- **MultipleValueString:** String field (see definition of “String” above) containing one or more space delimited values.• Country: String field (see definition of “String” above) representing a country using ISO 3166 Country code (2 character) values.
- **Currency:** String field (see definition of “String” above) representing a currency type using ISO 4217 Currency code (3 character) values.
- **Exchange:** String field (see definition of “String” above) representing a market or exchange.

- **month-year:** String field representing month of a year in YYYYMM format. Valid values: YYYY = 0000-9999, MM = 01-12.

- **UTCTimestamp:** Time/date combination represented in UTC (Universal Time Coordinated, also known as "GMT") in either YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss (milliseconds) format, colons, dash, and period required.

  Valid values:
  YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second) (without milliseconds).
  YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds).

- **Leap Seconds**: Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has never utilized these dates. During a leap second insertion, a UTCTimestamp field may read "19981231-23:59:59", "19981231-23:59:60", "19990101-00:00:00". (see http://tycho.usno.navy.mil/leapsec.html)

- **UTCTimeOnly:** Time-only represented in UTC (Universal Time Coordinated, also known as "GMT") in either HH:MM:SS (whole seconds) or HH:MM:SS.sss (milliseconds) format, colons, and period required.

  Valid values:
  HH = 00-23, MM = 00-60 (60 only if UTC leap second), SS = 00-59. (without milliseconds)
  HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss=000-999 (indicating milliseconds).

- **LocalMktDate:** Date of Local Market (vs. UTC) in YYYYMMDD format. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31.

- **UTCDate:** Date represented in UTC (Universal Time Coordinated, also known as "GMT") 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. The length field should specify the number of bytes of the value of the data field (up to but not including the terminating SOH). Caution: the value of one of these fields may contain the delimiter (SOH) character. Note that the value specified for this field should be followed by the delimiter (SOH) character as all fields are terminated with an "SOH".

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 monitors the status of the communication link and identifies 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). The HeartBtInt value should be agreed upon by the two firms and specified by the Logon initiator and echoed back by the Logon acceptor. Note that the same HeartBtInt value is used by both sides, the Logon “initiator” and Logon “acceptor”.

Ordered Message Processing:

The FIX protocol assumes complete ordered delivery of messages between parties. Implementers 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, the application could ignore messages 3 through 5 and generate a resend request for messages 2 through 5, or, preferably 2 through 0 (where 0 represents infinity). Another option would involve saving messages 3 through 5 and resending only message 2. In both cases, messages 3 through 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 message is generated. The message 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 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 possible duplicate message are the CheckSum, OrigSendingTime, SendingTime, BodyLength and PossDupFlag. Fields related to encryption (SecureDataLen and SecureData) may also require recasting.*

Possible Resends:

Ambiguous application level messages may be resent with the PossResend flag set. This is useful when 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 possible 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:

The integrity of message data content can be verified in two ways: verification of message length and a simple checksum of characters.

The message 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.

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 message 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, allocations, etc. require specific application level responses, executions can be rejected with the DK message but do not require explicit acceptance. See “Volume 1 - Business Message Reject” for details regarding the appropriate response message to specific application level messages.

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 but not required that all fields within the message body be encrypted. If repeating groups are used within a message and encryption is applied to part of the repeating group, then the entire repeating group must 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 previously agreed upon encryption methodology is declared in the *Logon* message. (For more detail on implementation of various encryption techniques see the application notes section on the FIX Web Site.)

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.

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, which are used as part of inter-firm communication. These tags can be registered/reserved via the FIX website.
The tag numbers greater than or equal to 10000 have been reserved for internal use (within a single firm) and do not need to be registered/reserved via the FIX website.

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. It is possible to maintain 24 hour connectivity and establish a new set of sequence numbers by sending a Logon message with the ResetSeqNumFlag set.
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.

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 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 previously agreed upon authentication method. If the initiator is successfully authenticated, the acceptor responds with a *Logon* message. If authentication fails, the session acceptor should shut down the connection after optionally sending a Logout message to indicate the reason of failure. Sending a Logout in this case is not required because doing so would consume a sequence number for that session, which in some cases may be problematic. The session initiator may begin to send messages immediately following the *Logon* message, however, the acceptor may not be ready to receive them. 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 *MsgSeqNum* field before sending any queued or new messages.** A comparison of the MsgSeqNum 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's MsgSeqNum to the next expected value. The section on message recovery later in this document deals with message gap handling.
- It is recommended to wait a short period of time following the Logon or to send a TestRequest and wait for a response to it before sending queued or new messages in order to allow both sides to handle resend request processing. Failure to do this could result in a ResendRequest message being issued by one's counterparty for each queued or new message sent.
- It is also recommended that an engine should store out of sequence messages in a temporary queue and process them in order when the gap is closed. This prevents generating resend requests for n->m, n->m+1, n->m+2, ... which can result in many resent PossDupFlag=Y messages.
- When using the ResetSeqNumFlag to maintain 24 hour connectivity and establish a new set of sequence numbers, the process should be as follows. Both sides should agree on a reset time and the party that will be the initiator of the process. Note that the initiator of the ResetSeqNum process may be different than the initiator of the Logon process. One side will initiate the process by sending a TestRequest and wait for a Heartbeat in response to ensure of no sequence number gaps. Once the Heartbeat has been received, the initiator should send a Logon with ResetSeqNumFlag set to Y and with MsgSeqNum of 1. The acceptor should respond with a Logon with ResetSeqNumFlag set to Y and with MsgSeqNum of 1. At this point new messages from either side should continue with MsgSeqNum of 2. It should be noted that once the initiator sends the Logon with the ResetSeqNumFlag set, the acceptor must obey this request and the message with the last sequence number transmitted "yesterday" may no longer be available. The connection should be shutdown and manual intervention taken if this process is initiated but not followed properly.

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. Session termination without receiving a Logout should treat the counterparty as logged out.

It is recommended that before sending the Logout message, a TestRequest should be issued to force a Heartbeat from the other side. This helps to ensure that there are no sequence number gaps.
Before actually closing the session, the Logout initiator should wait for the opposite side to respond with a confirming Logout message. This gives the acceptor a chance to perform any Gap Fill operations that may be necessary. Once the messages from the ResendRequest have been received, the acceptor should issue the Logout. The session may be terminated if the acceptor does not respond in an appropriate timeframe.
Note: 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 middle 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' at the beginning of the FIX session. Each message 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. Note that the SeqReset-Reset message (used only to recover from a disaster scenario vs. normal resend request processing) is an exception to this rule as it should be processed without regards to its MsgSeqNum. **If the incoming message has a sequence number less than expected and the PossDupFlag is not set, it indicates a serious error. 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 with PossDupFlag set to "Y"* message to replace the retransmission of administrative and application messages
3. Issue a *SeqReset-Reset with PossDupFlag set to "Y"* 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*, *ResendRequest*, *Heartbeat*, *TestRequest* and *SeqReset-Reset and SeqReset-GapFill*. The *SeqReset-GapFill* can also be used to skip application messages that the sender chooses not to retransmit (e.g. aged orders). This leaves Reject as the only administrative message which can be resent.

All FIX implementations must monitor incoming messages to detect inadvertently retransmitted administrative messages (*PossDupFlag* 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 *ResendRequest)*.

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 *NewSeqNo* field of the GapFill message contains the sequence number of the highest administrative message in this group plus 1.** For example, during a Resend operation there are 7 sequential administrative messages 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 is set to 9 because the remote side is expecting that as the next sequence number.** The *NewSeqNo* 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 when the incoming sequence number is greater than the expected incoming sequence number.

NOTE: In *ALL* cases except the Sequence Reset - Reset message, the FIX session should be terminated if the incoming sequence number is less than expected and the PossDupFlag 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* message. 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 *NewSeqNo* 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.

Messages Protocol

Standard Message header

Each administrative or application message is preceded by a standard header. The header identifies the message type, length, destination, sequence number, origination point and time.
Two fields help with resending messages. The PossDupFlag is set to Y 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 to Y when reissuing a message with a new sequence number (e.g. resending an order). The receiving application should process these messages as follows:
PossDupFlag - 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).

Message Routing Details – One Firm-to-One Firm (point-to-point)

The following table provides examples regarding the use of SenderCompID, TargetCompID, DeliverToCompID, and OnBehalfOfCompID when using a single point-to-point FIX session between two firms. Assumption (A=sellside, B =buyside):

	SenderCompID	OnBehalfOfCompID	TargetCompID	DeliverToCompID
A to B directly	A		B	
B to A directly	B		A	

The standard message header format is as follows:

Standard Message Header

Tag	Field Name	Req'd	Comments
8	BeginString	Y	FIX.4.3
9	BodyLength	Y	Automatically calculated by your FIX engine.
34	MsgSeqNum	Y	Automatically populated. You have to reset this Sequence Number before starting the connection the morning. Incrementation of the Sequence Number for each message exchanged
35	MsgType	Y	Depend on the type of message – Check the general FIX Protocol to populate this field
49	SenderCompID	Y	ULB_XXX_N XXX 3 characters to identify the client N: Number from 1 to Number of FIX Plug-ins
52	SendingTime	Y	Timestamp with following format : 20060620-10:03:06
56	TargetCompID	Y	ULB_NEWEDGE_XXX _N XXX 3 characters to identify the client N: Number from 1 to Number of FIX Plug-ins

43	PossDupFlag	N	Always required for retransmitted messages, whether prompted by the sending system or as the result of a resend request.
57	TargetSubID	N	“ADMIN” reserved for administrative messages not intended for a specific user.
97	PossResend	N	Required when message may be duplicate of another message sent under a different sequence number.
122	OrigSendingTime	N	Required for message resent as a result of a ResendRequest. If data is not available set to same value as SendingTime

Standard 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 standard message trailer format is as follows:

Standard Message Trailer

Tag	Field Name	Req'd	Comments
10	CheckSum	Y	Automatically populated by your FIX engine

Administrative Message body

The administrative messages address the utility needs of the protocol. The following section describes each message and provides the message layout.
Administrative messages will be generated from both sides of the connection.

Heartbeat

The Heartbeat monitors the status of the communication link and identifies 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

The logon message authenticates a user establishing 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 (same value used by both sides). The HeartBtInt value should be agreed upon by the two firms and specified by the Logon initiator and echoed back by the Logon acceptor.
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 logon format is as follows:

Logon

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = A
98	EncryptMethod	Y	0 = None
108	HeartBtInt	Y	Note same value used by both sides We propose 30 seconds
141	ResetSeqNumFlag	N	Indicates both sides of a FIX session should reset sequence numbers
	Standard Trailer	Y	

Test Request

The test request message forces a heartbeat from the opposing application. The test request message checks sequence numbers or verifies communication line status. The opposite application responds to the Test Request with a Heartbeat containing the TestReqID.
The TestReqID verifies that the opposite application is generating the heartbeat as the result of Test Request and not a normal timeout. The opposite application includes 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, or, preferably, 7-0 (0 represents infinity). This latter approach is strongly recommended to recover from out of sequence conditions as it allows for faster recovery in the presence of certain race conditions when both sides are simultaneously attempting to recover a gap.

- 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 = 0 (represents infinity) .

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 (session-level)

The reject message should be issued when a message is received but cannot be properly processed due to a session-level rule violation. 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: The receiving application should disregard any message that is garbled, cannot be parsed or fails a data integrity check. 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 indicates a serious error that 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 and sent with PossResend=Y.

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

If an application-level message received fulfills session-level rules, it should then be processed at a business message-level. If this processing detects a rule violation, a business-level reject should be issued. Many business-level messages have specific "reject" messages, which should be used.

The reject format is as follows:

Reject

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = 3
45	RefSeqNum	Y	MsgSeqNum of rejected message
58	Text	N	Human readable of reason for rejection
371	RefTagID	N	The tag number of the FIX field being referenced.
372	RefMsgType	N	The MsgType of the FIX message being referenced.
373	SessionRejectReason	N	Code to identify reason for a session-level Reject message. 0 = Invalid tag number 1 = Required tag missing 2 = Tag not defined for this message type 3 = Undefined Tag 4 = Tag specified without a value 5 = Value is incorrect (out of range) for this tag 6 = Incorrect data format for value 7 = Decryption problem 8 = Signature problem 9 = CompID problem 10 = SendingTime accuracy problem 11 = Invalid MsgType (Note other session-level rule violations may exist in which case SessionRejectReason is not specified)
	Standard Trailer	Y	

Session level reject Sample Message

	Cancel request
Standard Header	8= FIX.4.4 9= 121 35= 3 49= ULB_NEWEDGE_XXX_1 56= ULB_XXX_1 34= 139 52= 20070906-17:02:36
Main body	45= 136 371= 41 372= F 373= 1 58= Required Tag(41) Missing
Standard Trailer	10= 017

Sequence Reset (Gap Fill)

The sequence reset message is used by the sending application to reset the incoming sequence number on the opposing side. This message has two modes: "Sequence Reset-Gap Fill" when GapFillFlag is 'Y' and "Sequence Reset-Reset" when GapFillFlag is N or not present. The "Sequence Reset-Reset" mode should **ONLY** be used to recover from a disaster situation which cannot be otherwise recovered via "Gap Fill" mode. 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 – Gap Fill is used to mark the place of that message.

During normal resend processing, a number of administrative messages are not resent, the Sequence Reset – Gap Fill 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 via the use of Sequence Reset - Reset

The sending application will initiate the sequence reset. **The message in all situations specifies NewSeqNo to reset as the value of the next sequence number immediately following the messages and/or sequence numbers being skipped**.

If the GapFillFlag 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. The MsgSeqNum in the header should be ignored (i.e. the receipt of a Sequence Reset - Reset message with an out of sequence MsgSeqNum should not generate resend requests**).**

Sequence Reset – Reset should NOT be used as a normal response to a Resend Request (use Sequence Reset – Gap Fill). The Sequence Reset – Reset should ONLY be used to recover from a disaster situation which cannot be recovered via the use of Sequence Reset – Gap Fill. Note that the use of Sequence Reset – Reset may result in the possibility of lost messages

If the GapFillFlag field is present (and equal to Y), the MsgSeqNum should conform to standard message sequencing rules (i.e. the MsgSeqNum of the Sequence Reset-GapFill message should represent the beginning MsgSeqNum in the GapFill range because the remote side is expecting that next message).

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. It is possible to have multiple ResendRequests issued in a row (i.e. 5 to 10 followed by 5 to 11). If sequence number 8, 10, and 11 represent application messages while the 5-7 and 9 represent administrative messages, the series of messages as result of the Resend Request may appear as SeqReset-GapFill with NewSeqNo of 8, message 8, SeqReset-GapFill with NewSeqNo of 10, and message 10. This could then followed by SeqReset-GapFill with NewSeqNo of 8, message 8, SeqReset-GapFill with NewSeqNo of 10, message 10, and message 11. One must be careful to ignore the duplicate SeqReset-GapFill which is attempting to lower the next expected sequence number. This can be detected by checking to see if its MsgSeqNum is less than expected. If so, the SeqReset-GapFill is a duplicate and should be discarded.

The Sequence Reset format is as follows:

Sequence Reset

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

Logout

The logout message initiates or confirms 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.
After sending the Logout message, the logout initiator should not send any messages unless requested to do so by the logout acceptor via a ResendRequest.
The logout format is as follows:

Logout

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

Application Message body

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.

Note that in all the following bodies, the Tag 1 (Account) should respect several rules:
1. The account Identifier is a string of 9 characters
2. The string should be in capital
3. The three first characters should identify the trader or the client
4. The five last characters are free to identify your account

All orders sent with an unknown account will be rejected

Example: XXXNWGLLL
Where LLL is the location of the NEWEDGE that manages the Client

Other comment: it is possible to trade other exchanges through the OMS – Please refer to chapter 19 to have more explanation

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 HandlInst field).
New Order messages received with the PossResend flag set in the header should be validated by ClOrdID. Implementations should also consider checking 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.
The value specified in the TransactTime field should allow the receiver of the order to apply business rules to determine if the order is potentially "stale" (e.g. in the event that there have been communication problems).
The format for the new order message is as follows:

New Order – Single

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = D
1	Account	Y	XXXNWGLLL
11	ClOrdID	Y	Unique identifier of the order as assigned by institution or by the intermediary (CIV term, not a hub/service bureau) with closest association with the investor. NEWEDGE suggests field has to be built with following subpart: client ID (3 characters) + Date (6 characters YYMMDD) + Increment number (4 characters) + 1 characters (R for a New Order, A for a Modification and C for a Cancel) Global max length = 14 (Instead of 16 (official length) due to NEWEDGE internal processes)
21	HandlInst	Y	Handling Instruction: 1 = automatic 2 = broker 3 = manual (care) Note: 2 behaves like automatic for the moment

38	OrderQty	Y	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified. *- Part of 'Instrument' component block -*
40	OrdType	Y	1 = Market 2 = Limit 3 = Stop 4 = Stop Limit A = At Best O = Open *
50	SenderSubID	Y	This field is mandatory only for Order message and should not be populated for Admin messages (Logon, HB…) In case of Order messages, this field should be NEWEDGE_XXX XXX 3 characters to identify the client
54	Side	Y	Should match original order's side, however, if bilaterally agreed to the following groups could potentially be interchanged: • Buy and Buy Minus • Sell, Sell Plus, Sell Short, and Sell Short Exempt Cross, Cross Short, and Cross Short Exempt Only Buy (1) and Sell (2) are supported
60	TransactTime	Y	Time this order request was initiated/released by the trader, trading system, or intermediary. TimeStamp with following format: 20060620-10:06:51
100	ExDestination	Y	Exchange code. Has to be agreed with both counter parties. You can find the value of each ExDestination on corresponding chapters below List of available ExDestination : XAMS XASX Australia Stock Exchange XBKK XBRU XCBT CBOT XCEC COMEX

			XCME CME XDMI XETR XEUR EUREX XHER XHKF Hong Kong Futures Exchange XHKG Stock Exchange Hong Kong XIDX XIPE ICE US XJAS XKFE Korean Futures Exchange XKLS XKOS Korea KOSDAQ XKRX Korea Exchange (Equities) XLIF LIFFE XMAT MATIF XMOO XNSE XNYF XNYM NYMEX XOSE Osaka Securities Exchange XPAR XPHS XSES Singapore SGX (Equities) XSFE Sydney Future Exchange XSIM Singapore SGX (Derivatives) XTAF Taiwan Fitires Exchange XTAI Taiwan Stock Exchange XTFF Tokyo TFX (aka TIFFE) XTKS Tokyo Stock Exchange XTKT Tokyo TOCOM
22	SecurityIDSource	C	Required if SecurityID is specified. 1 = CUSIP Code 2 = SEDOL Code 3 = QUIK Code 4 = ISIN Code 5 = RIC Code *- Part of 'Instrument' component block -* *
44	Price	C	Required for limit OrdTypes. For F/X orders, should be the "all-in" rate (spot rate adjusted for forward points). Can be used to specify a limit price for a pegged order, previously indicated, etc.

48	SecurityID	C	Takes precedence in identifying security to counterparty over SecurityAltID block. Requires SecurityIDSource if specified. *- Part of 'Instrument' component block -* *
55	Symbol	C	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists The value must correspond to an existing value in the destination system (see data dictionary) *- Part of 'Instrument' component block -* *
99	StopPx	C	Required for OrdType = "Stop" or OrdType = "Stop limit".
111	MaxFloor	C	Maximum number of shares within an order to be shown on the exchange floor at any given time
114	LocateReqd	C	Required for short sell orders
126	ExpireTime	C	Conditionally required if TimeInForce = GTD and ExpireDate is not specified.
167	SecurityType	C	It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. Futures and Options should be specified using the CFICode[461] field instead of SecurityType[167] (Refer to Volume 7 – Recommendations and Guidelines for Futures and Options Markets.") *- Part of 'Instrument' component block -* *
200	MaturityMonthYear	C	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified. e.g.: 201006 *- Part of 'Instrument' component block -*
202	StrikePrice	C	Used for derivatives, such as options and covered warrants *- Part of 'Instrument' component block -*
210	MaxShow	C	Conditional: used for Iceberg order type
461	CFICode	C	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *- Part of 'Instrument' component block -* *

541	MaturityDate	C	Specifies date of maturity (a full date). Note that standardized derivatives which are typically only referenced by month and year (e.g. S&P futures).may use MaturityMonthYear and/or this field. When using MaturityMonthYear, it is recommended that markets and sell sides report the MaturityDate on all outbound messages as a means of data enrichment. *- Part of 'Instrument' component block -*
15	Currency	N	Identifies currency used for Price. Absence of this field is interpreted as the default for the security
18	ExecInst	N	Can contain multiple instructions, space delimited. If OrdType=P, exactly one of the following values (ExecInst = L, R, M, P, O, T, or W) must be specified.
58	Text	N	Free format text string
59	TimeInForce	N	Absence of this field indicates Day order 0 = Day 1 = Good Till Cancel (GTC) 2 = At Opening 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK) 6 = Good To Date (GTD) S = Session *
77	PositionEffect	N	For use in derivatives omnibus accounting
110	MinQty	N	Minimum quantity of an order to be executed
203	CoveredOrUncovered	N	For use with derivatives, such as options
229	TradeOriginationDate	N	
377	SolicitedFlag	N	
432	ExpireDate	N	Conditionally required if TimeInForce = GTD and ExpireTime is not specified.
526	SecondaryClOrdID	N	
528	OrderCapacity	N	
529	OrderRestrictions	N	
582	CustOrderCapacity	N	
142	SenderLocationID (String)	Y	Mandatory for CME exchanges only, should be the country two characters. e.g. HK, JP
1028	ManualOrderIndicator (Char)	Y	Mandatory for CME exchanges only, indicates if the order is automatically generated (N) or manually (Y).
	Standard Trailer	Y	

* Details of these fields are given by Market further on the document

New Order Message Samples Per Order Type

Standard Header	8= FIX.4.4 9= 173 35= D 49= ULB_APR_1 56= ULB_NEWEDGE_APR_1 34= 221 52= 20041014-19:30:23								
Common tags	1= TESTACCOUNT 11= APP000007382 21= 1 38= 5 50= NEWEDGE-X 54= 2 60= 20041014-19:30:23 100= MATIF 200= 200412 55= JFCE 461= FXXXXX								
	Market (day)	**Market on Open**	**Limit (day)**	**Limit FOK**	**Limit GTC**	**Limit GTD**	**Limit IOC**	**Stop (day)**	**Stop-Limit (day)**
Specific tags	**40=1**	**40=1** **59=2**	**40=2** **44=3712**	**40=2** **44=3712** **59=4**	**40=2** **44=3712** **59=1**	**40=2** **44=3712** **59=6** **126=20041201-00:00:00**	**40=2** **44=3712** **59=3**	**40=3** **99=3712**	**40=4** **44=3713** **99=3712**
Standard Trailer	10= 011								

New Order Message Samples Per Security Type

Standard Header	8= FIX.4.4 9= 173 35= D 49= ULB_APR_1 56= ULB_NEWEDGE_APR_1 34= 221 52= 20050428-22:30:23			
Common tags	1= TESTACCOUNT 11= APP000007382 21= 1 38= 5 50= APP 54= 2 60= 20050428-22:30:23 200= 200506 40=1			
	Futures order: **Sell 5 MATIF JFCE @ Market**	**Option order:** **Sell 5 CBOT OZN 130 @ Market**	**Equity order:** **Sell 5 AMEX IBM @ Market**	**Equity Option order** **Sell 5 ISE IBM 60 @ Market**
Specific tags	100= MATIF 55= JFCE 461= FXXXXX	100= CBOT 55= OZN 461= OPXXXX 202= 130	100= AMEX 55= IVX 167= CS	100= ISE 55= IBM 461= OCXXXX 202= 60 203= 0 204= 0 77= O

Standard Trailer	10= 011

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 on working orders
5. reject orders

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 report contains three fields which are used to communicate both the current state of the order as understood by the broker and the purpose of the message: OrdStatus, ExecType and ExecTransType.

In an execution report the OrdStatus is used to convey the current state of the order. If an order simultaneously exists in more than one order state, the value with highest precedence is the value that is reported in the OrdStatus field. The order statuses are as The ExecType is used to identify the purpose of the execution report message. To transmit a change in OrdStatus for an order, the broker(sell side) should send an Execution Report with the new OrdStatus value in both the ExecType AND the OrdStatus fields to signify this message is changing the state of the order. The only exception to this rule is that when rejecting a cancel or cancel/replace request the CancelReject message is used both to reject the request and to communicate the current OrdStatus. An ExecType of Pending Cancel or Pending Replace is used to indicate that a cancel or cancel/replace request is being processed. An ExecType of Canceled or Replace is used to indicate that the cancel or cancel/replace request has been successfully processed.

Execution information (e.g. new partial fill or complete fill) should not be communicated in the same report as one which communicates other state changes (such as pending cancel, pending replace, canceled, replaced, accepted, done for day etc).

Any fills which occur and need to be communicated to the customer while an order is "pending" and waiting to achieve a new state (i.e. via a Order Cancel Replace (aka Order Modification) Request) must contain the "original" (current order prior to state change request) order parameters (i.e. ClOrdID, OrderQty, Price, etc). These fills will cause the CumQty and AvgPx to be updated. An order cannot be considered replaced until it has been explicitly accepted and confirmed to have reached the replaced status via an execution report with ExecType = 'Replace', at which time the effect of the replacement (ClOrdID, new quantity or limit price etc) will be seen. Note that due to the precedence rules above, in reports where ExecType=Replace, OrdStatus may not = Replaced.

Requests to cancel or cancel/replace an order are only acted upon when there is an outstanding order quantity. Requests to replace the OrderQty to a level less than the CumQty will be interpreted by the broker as requests to stop executing the order. Requests to change price on a filled order will be rejected (see Order Cancel Reject message type). The OrderQty, CumQty, LeavesQty, and AvgPx fields should be calculated to reflect the cumulative result of all versions of an order. For example, if partially filled order A were replaced by order B, the OrderQty, CumQty, LeavesQty, and AvgPx on order B's fills should represent the cumulative result of order A plus those on order B.

The general rule is: OrderQty = CumQty + LeavesQty.

The execution report message format is as follows:

Execution Report

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = 8
1	Account	Y	Required for executions against electronically submitted orders *which were assigned an account by the institution or intermediary* XXXNWGLLL
6	AvgPx	Y	Calculated Average Price of all fills of this Order
11	ClOrdID	Y	Required for executions against electronically submitted orders which were assigned an ID by the institution or intermediary. Not required for orders manually entered by the broker or fund manager (for CIV orders).
14	CumQty	Y	Currently executed quantity for chain of orders.
17	ExecID	Y	Unique identifier of execution message as assigned by sell-side (broker, exchange, ECN) (will be 0 (zero) forExecType=I (Order Status)).
37	OrderID	Y	OrderID is required to be unique for each chain of orders.
38	OrderQty	Y	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified. *- Part of 'Instrument' component block -*
39	OrdStatus	Y	Describes the current state of a CHAIN of orders, same scope as OrderQty, CumQty, LeavesQty, and AvgPx Valid values: 0 = New 1 = Partially filled 2 = Filled 3 = Done for day 4 = Canceled ~~5 = Replaced~~ **(Removed/Replaced)** 6 = Pending Cancel (e.g. result of Order Cancel Request) 7 = Stopped 8 = Rejected 9 = Suspended A = Pending New B = Calculated C = Expired D = Accepted for bidding E = Pending Replace (e.g. result of Order Cancel/Replace Request)

			Value 5 removed *
54	Side	Y	1 = Buy 2 = Sell 3 = Buy minus (US markets only) 4 = Sell plus (US markets only) 5 = Sell short (US markets only) 6 = Sell short exempt (US markets only)

150	ExecType	Y	Describes the purpose of the execution report. Describes the specific ExecutionRpt (i.e. Pending Cancel) while OrdStatus will always identify the current order status (i.e. Partially Filled) Valid values: 0 = New ~~1 = Partial fill~~ **(Replaced)** ~~2 = Fill~~ **(Replaced)** 3 = Done for day 4 = Canceled 5 = Replace 6 = Pending Cancel (e.g. result of Order Cancel Request) 7 = Stopped 8 = Rejected 9 = Suspended A = Pending New B = Calculated C = Expired D = Restated (ExecutionRpt sent unsolicited by sellside, with ExecRestatementReason set) E = Pending Replace (e.g. result of Order Cancel/Replace Request) F = Trade (partial fill or fill) G = Trade Correct (formerly an ExecTransType) H = Trade Cancel (formerly an ExecTransType) I = Order Status (formely an ExecTransType) ExecType 1 & 2 replaced by F *
151	LeavesQty	Y	Quantity open for further execution. If the OrdStatus is Canceled, DoneForTheDay, Expired, Calculated, or Rejected (in which case the order is no longer active) then LeavesQty could be 0, otherwise LeavesQty = OrderQty - CumQty.
19	ExecRefID	C	Required for Trade Cancel and Trade Correct ExecType messages
22	SecurityIDSource	C	Required if SecurityID is specified. 1 = CUSIP Code 2 = SEDOL Code 3 = QUIK Code 4 = ISIN Code 5 = RIC Code *- Part of 'Instrument' component block -*

41	OrigClOrdID	C	Conditionally required for response to an electronic Cancel or Cancel/Replace request (ExecType=PendingCancel, Replace, or Canceled). ClOrdID of the previous accepted order (NOT the initial order of the day) when canceling or replacing an order.
48	SecurityID	C	Takes precedence in identifying security to counterparty over SecurityAltID block. Requires SecurityIDSource if specified. *- Part of 'Instrument' component block -* *
55	Symbol	C	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists The value must correspond to an existing value in the destination system (see data dictionary) *- Part of 'Instrument' component block -* *
66	ListID	C	Required for executions against orders which were submitted as part of a list.
110	MinQty	C	Minimum quantity of an order to be executed
126	ExpireTime	C	Conditionally required if TimeInForce = GTD and ExpireDate is not specified.
167	SecurityType	C	It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. Futures and Options should be specified using the CFICode[461] field instead of SecurityType[167] (Refer to Volume 7 – Recommendations and Guidelines for Futures and Options Markets.") *- Part of 'Instrument' component block -* *
200	MaturityMonthYear	C	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified. *- Part of 'Instrument' component block -*
210	MaxShow	C	Conditional: used for Iceberg order type
202	StrikePrice	C	Used for derivatives, such as options and covered warrants *- Part of 'Instrument' component block -*
111	MaxFloor	C	Maximum number of shares within an order to be shown on the exchange floor at any given time
378	ExecRestatementReason	C	Required for ExecType = D (Restated).

461	CFICode		C	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *- Part of 'Instrument' component block -* *
541	MaturityDate		C	Specifies date of maturity (a full date). Note that standardized derivatives which are typically only referenced by month and year (e.g. S&P futures).may use MaturityMonthYear and/or this field. When using MaturityMonthYear, it is recommended that markets and sell sides report the MaturityDate on all outbound messages as a means of data enrichment. *- Part of 'Instrument' component block -*
555	NoLegs		C	Number of legs
→	**600**	**LegSymbol**	C	Equivalent to the Symbol (55) for New Order - Single
→	**624**	**LegSide**	C	Specific to the <InstrumentLeg> (not in <Instrument>)
→	**623**	**LegRatioQty**	C	Specific to the <InstrumentLeg> (not in <Instrument>)
→	***566***	***LegPrice***	C	Provide only if a Price is required for a specific leg. Used for anchoring the overall multileg security price to a specific leg Price.
→	***637***	***LeglastPx***	C	Used to report the execution price assigned to the leg of the multileg instrument
15	Currency		N	Identifies currency used for Price. Absence of this field is interpreted as the default for the security
21	HandlInst		N	Handling Instruction: 1 = automatic 2 = broker 3 = manual (care) Note: 2 behaves like automatic for the moment
29	LastCapacity		N	
30	LastMkt		N	Market of execution for last fill
31	LastPx		N	Price of this (last) fill. Required if ExecType = Trade or Trade Correct. Should represent the "all-in" (LastSpotRate + LastForwardPoints) rate for F/X orders.). If ExecType=Stopped, represents the price stopped/guaranteed/protected at.

32	LastQty	N	Quantity (e.g. shares) bought/sold on this (last) fill. Required if ExecType = Trade or Trade Correct. If ExecType=Stopped, represents the quantity stopped/guaranteed/protected for.
40	OrdType	N	1 = Market 2 = Limit 3 = Stop 4 = Stop Limit A = At Best O = Open
44	Price	N	Required if specified on the order
58	Text	N	
59	TimeInForce	N	Absence of this field indicates Day order 0 = Day 1 = Good Till Cancel (GTC) 2 = At Opening 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK) 6 = Good To Date (GTD) S = Session
60	TransactTime	N	Time the transaction represented by this ExecutionReport occurred
77	PositionEffect	N	For use in derivatives omnibus accounting
99	StopPx	N	Required if specified on the order
103	OrdRejReason	N	For optional use with ExecType = 8 (Rejected)
198	SecondaryOrderID	N	Can be used to provide order id used by exchange or executing system.
336	TradingSessionID	N	
432	ExpireDate	N	Conditionally required if TimeInForce = GTD and ExpireTime is not specified.
442	MultiLegReportingType	N	Default is a single security if not specified.
528	OrderCapacity	N	
529	OrderRestrictions	N	
582	CustOrderCapacity	N	
	Standard Trailer	*Y*	

* See Appendix B “Replaced Features and Supported Approach”

Standard Header	8=	FIX.4.4
	9=	230
	35=	8
	49=	ULB_APR_1
	56=	ULB_NEWEDGE_APR_1
	34=	119

	52= 20041021-14:52:03					
	Order Active	**Partial fill**	**Full Execution**	**Replaced**	**Canceled**	**Reject**
Specific Required tags	39=0 150=0	39=1 150=F	39=2 150=F	39=5 150=5	39=4 150=4	39=8 150=8
Common required tags	1=TESTACCNT 6=0 11=APP0007561 14=0 17=04000104 37=309E140K04 38=1 54=1 55=ES 151=1	1=TESTACCNT 6=111275.000000 11=APP0007444 14=1 17=32 37=20041018001341 38=5 54=2 55=ES 151=4		1=TESTACCNT 6=0 11=APP0007273 14=0 17=1170 37=RE12237 38=5 54=2 55=FGBL 151=0		
Common tags	30=CME 31=0 32=0 40=2 50=NEWEDGE-X 59=3 60=20041021-13:58:29 461=FXXXXX 200=200412	30=CME 31=111275.00 32=1 40=1 50=NEWEDGE-X 59=6 60=20041018-10:18:48 461=FXXXXX 200=200412	30=CME 31=111250.00 32=3 40=1 50=NEWEDGE-X 59=6 60=20041018-10:18:48 461=FXXXXX 200=200412	30=XEUR 31=0 32=0 40=3 50=NEWEDGE-X 59=0 60=20041012-16:44:52 461=FXXXXX 200=200412	30=XEUR 31=0 32=0 40=2 50=NEWEDGE-X 59=4 60=20041012-17:33:53 461=OPXXXX 200=200412	30=XEUR 31=0 32=0 40=3 50=NEWEDGE-X 59=0 60=20041012-16:20:58 461=FXXXXX 200=200412
Specific tags	41=APP0007216 44=6			99=111	44=3450 202=3500	58=PRICE HIGHER THAN INSIDE MKT ASK PRC 99=115.45 103=RE12237
Standard Trailer	10=048					

Order Cancel Request

The order cancel request message requests the cancelation of **all** of the remaining quantity of an existing order. Note that the Order Cancel/Replace Request should 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.

A cancel request is assigned a ClOrdID and is treated as a separate entity. If rejected, the ClOrdID of the cancel request will be sent in the Cancel Reject message, as well as the ClOrdID of the actual order in the OrigClOrdID field. The ClOrdID assigned to the cancel request must be unique amongst the ClOrdID 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*
1	Account	Y	XXXNWGLLL
11	ClOrdID	Y	Unique ID of cancel request as assigned by the institution.

41	OrigClOrdID	Y	ClOrdID of the previous non-rejected order (NOT the initial order of the day) when canceling or replacing an order. For NEWEDGE his field has to be built with following subpart: XXX 3 characters to identify the client + Date (6 characters YYMMDD) + Increment number (6 characters) + 1 characters (R for a New Order, A for a Modification and C for a Cancel)
50	SenderSubID	Y	This field is mandatory only for Order message and should not be populated for Admin messages (Logon, HB…) In case of Order messages, this field should be NEWEDGE_XXX XXX 3 characters to identify the client
54	Side	Y	Should match original order's side, however, if bilaterally agreed to the following groups could potentially be interchanged: • Buy and Buy Minus • Sell, Sell Plus, Sell Short, and Sell Short Exempt Cross, Cross Short, and Cross Short Exempt
60	TransactTime	Y	Time this order request was initiated/released by the trader or trading system. TimeStamp with following format: 20060620-10:06:51
22	SecurityIDSource	C	**Error! Reference source not found.** *- Part of 'Instrument' component block -* *
48	SecurityID	C	Takes precedence in identifying security to counterparty over SecurityAltID block. Requires SecurityIDSource if specified. *- Part of 'Instrument' component block -* *
55	Symbol	C	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists The value must correspond to an existing value in the destination system (see data dictionary) *- Part of 'Instrument' component block -* *
66	ListID	C	Required for List Orders

167	SecurityType	C	It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. Futures and Options should be specified using the CFICode[461] field instead of SecurityType[167] (Refer to Volume 7 – Recommendations and Guidelines for Futures and Options Markets.") *- Part of 'Instrument' component block -* *
200	MaturityMonthYear	C	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified. *- Part of 'Instrument' component block -*
202	StrikePrice	C	Used for derivatives, such as options and covered warrants *- Part of 'Instrument' component block -*
461	CFICode	C	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *- Part of 'Instrument' component block -* *
541	MaturityDate	C	Specifies date of maturity (a full date). Note that standardized derivatives which are typically only referenced by month and year (e.g. S&P futures).may use MaturityMonthYear and/or this field. When using MaturityMonthYear, it is recommended that markets and sell sides report the MaturityDate on all outbound messages as a means of data enrichment. *- Part of 'Instrument' component block -*
37	OrderID	N	Unique identifier of most recent order as assigned by sell-side (broker, exchange, ECN).
38	OrderQty	N	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified.
58	Text	N	
581	AccountType	N	
142	SenderLocationID (String)	Y	Mandatory for CME exchanges only, should be the country two characters. e.g. HK, JP
1028	ManualOrderIndicator (Char)	Y	Mandatory for CME exchanges only, indicates if the order is automatically generated (N) or manually (Y).
	Standard Trailer	*Y*	

* Details of these fields are given by Market further on the document

Example of complete FIX message – Single Leg

Instrument

Future
The instrument code which is use to send on Order on SGX is the RIC code.
Different information are required by the system to interpret the instrument: information about the type of contract and information that composed the Generic code
These information are:

- The CFI Code: this allows identifying the type of the instrument (FXXXXX for Future and OXXXXX for Option, MXXXXX for a Strategy)
- The Symbol: the Symbol of the instrument should be the RIC Code of the contract (We can provide a list of available codes)
- The Maturity of the Instrument (with 6 characters ex: 200509 for September 2005)

Option
As for Future, the instrument code for Options is the RIC Code but there are several fields to add.
These information are:

- The CFI Code: this allows identifying the type of the instrument (FXXXXX for Future and OXXXXX for Option, MXXXXX for a Strategy)
- The Symbol: the Symbol of the instrument should be the RIC Code of the contract (We can provide a list of available codes)
- The Maturity of the Instrument (with 6 characters ex: 200509 for September 2005)
- The Type of Option (Put or Call) is populated on the second character of the CFICode (OPXXXX for an Option Put and OCXXXX for an Option Call)
- The Strike Price.

Price Format

Future
The price format that is used to send an order for Future on SGX is directly the price format of the market (in decimal) – there are no particular rules on this market.

Option
There is no particular Price Format for Options on SGX

Side

1=Buy
2=Sell
3 = Buy minus (US markets only)

Buy 25 NK;U05 @ 12260

8=FIX.4.3|9=223|35=D|49=ULB_SMA_2|56=ULB_NEWEDGE_SMA_2|34=691|50=ULlink|52=20050826-14:19:31|11=SMA000019141|1=ULlink|21=1|100=SGX|55=NK|48=NK|22=5|461=FXXXXX|200=200509|**54=1**|60=20050826-14:19:31|38=25|40=2|44=12260|15=JPY|59=0|10=189|

Example of complete FIX message – Multi-Leg

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. Filled orders cannot be changed (i.e quantity reduced or price change. However, the broker/sellside may support increasing the order quantity on a currently filled order).

When rejecting a Cancel/Replace Request, the Cancel Reject message should provide the ClOrdID and OrigClOrdID values which were specified on the Cancel/Replace Request message for identification

The execution message responds 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
11	ClOrdID	Y	Unique order id assigned by institution or by the intermediary with closest association with the investor. to the cancel request or to the *replacement* order.
37	OrderID	Y	If CxlRejReason=”Unknown order”, specify “NONE”.
39	OrdStatus	Y	OrdStatus value after this cancel reject is applied.
41	OrigClOrdID	Y	ClOrdID which could not be canceled/replaced. ClOrdID of the previous accepted order (NOT the initial order of the day) when canceling or replacing an order.
434	CxlRejResponseTo	Y	
66	ListID	C	Required for rejects against orders which were submitted as part of a list.
1	Account	N	XXXNWGLLL
58	Text	N	
60	TransactTime	N	
102	CxlRejReason	N	
198	SecondaryOrderID	N	Can be used to provide order id used by exchange or executing system.
	Standard Trailer	*Y*	

Cancel Reject Message Sample

	Cancel reject	
Standard Header	8=	FIX.4.4
	9=	161
	35=	9
	49=	ULB_APR_1
	56=	ULB_NEWEDGE_APR_1
	34=	291
	52=	20041006-16:15:25
Main body	11=	CxlId000007217
	37=	08388674568641012
	41=	APP00007216
	39=	2
	434=	1
	102=	2
	58=	Order not present
Standard Trailer	10=	057

Order Cancel/Replace Request (a.k.a. Order Modification 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.
Cancel/Replace will be used to change any valid attribute of an open order (i.e. reduce/increase quantity, change limit price, change instructions, etc.) It can be used to re-open a filled order by increasing OrderQty.
An immediate response to this message is required. It is recommended that an ExecutionRpt with ExecType=Pending Replace be sent unless the Order Cancel/Replace Request can be immediately accepted (ExecutionRpt with ExecType=Replaced) or rejected (Order Cancel Reject message).

The Cancel/Replace request will only be accepted if the order can successfully be pulled back from the exchange floor without executing. Requests which cannot be processed will be rejected using the Cancel Reject message. The Cancel Reject message should provide the ClOrdID and OrigClOrdID values which were specified on the Cancel/Replace Request message for identification.
Note that while it is necessary for the ClOrdID to change and be unique, the broker's OrderID field does not necessarily have to change as a result of the Cancel/Replace request.
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. The fields which can be changed via this message are:

OrderQty	ExpireTime
OrdType	MinQty
Price	MaxFloor
TimeInForce	StopPx

The format of the Order Cancel/Replace Request message is:

Order Cancel/Replace Request (a.k.a. Order Modification Request)

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = G
1	Account	Y	XXXNWGLLL
11	ClOrdID	Y	Unique identifier of *replacement* order as assigned by institution or by the intermediary with closest association with the investor.. Note that this identifier will be used in ClOrdID field of the Cancel Reject message if the replacement request is rejected.
21	HandlInst	Y	Handling Instruction: 1 = automatic 2 = broker 3 = manual (care) Note: 2 behaves like automatic for the moment
38	OrderQty	Y	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified. *- Part of 'Instrument' component block -*
40	OrdType	Y	1 = Market 2 = Limit 3 = Stop 4 = Stop Limit A = At Best O = Open *
41	OrigClOrdID	Y	Unique identifier of the order as assigned by institution or by the intermediary (CIV term, not a hub/service bureau) with closest association with the investor. NEWEDGE suggests field has to be built with following subpart: client ID (3 characters) + Date (6 characters YYMMDD) + Increment number (4 characters) + 1 characters (R for a New Order, A for a Modification and C for a Cancel) Global max length = 14 (Instead of 16 (official length) due to NEWEDGE internal processes)
50	SenderSubID	Y	This field is mandatory only for Order message and should not be populated for Admin messages (Logon, HB, …) In case of Order messages, this field should be

			NEWEDGE_XXX XXX 3 characters to identify the client
54	Side	Y	Should match original order's side, however, if bilaterally agreed to the following groups could potentially be interchanged: • Buy and Buy Minus • Sell, Sell Plus, Sell Short, and Sell Short Exempt Cross, Cross Short, and Cross Short Exempt
60	TransactTime	Y	Time this order request was initiated/released by the trader or trading system.
22	SecurityIDSource	C	Identifies class or source of the SecurityID (48) value. Required if SecurityID is specified. Values supported: 1 = CUSIP 2 = SEDOL 3 = QUIK 4 = ISIN number 5 = RIC code 6 = ISO Currency Code 7 = ISO Country Code 8 = Exchange Symbol 9 = Consolidated Tape Association (CTA) Symbol (SIAC CTS/CQS line format) A = Bloomberg Symbol B = Wertpapier C = Dutch D = Valoren E = Sicovam F = Belgian G = "Common" (Clearstream and Euroclear) H = Clearing House / Clearing Organization I = ISDA/FpML Product Specification J = Options Price Reporting Authority 100+ are reserved for private security identifications. Part of Instrument component block.- *Part of 'Instrument' component block -* *
44	Price	C	Required for limit OrdTypes. For F/X orders, should be the "all-in" rate (spot rate adjusted for forward points). Can be used to specify a limit price for a pegged order, previously indicated, etc.

48	SecurityID	C	Takes precedence in identifying security to counterparty over SecurityAltID block. Requires SecurityIDSource if specified. *- Part of 'Instrument' component block -* *
55	Symbol	C	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists The value must correspond to an existing value in the destination system (see data dictionary) *- Part of 'Instrument' component block -* *
66	ListID	C	Required for List Orders
99	StopPx	C	Required for OrdType = "Stop" or OrdType = "Stop limit".
110	MinQty	C	Minimum quantity of an order to be executed
114	LocateReqd	C	
126	ExpireTime	C	Conditionally required if TimeInForce = GTD and ExpireDate is not specified.
167	SecurityType	C	It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. Futures and Options should be specified using the CFICode[461] field instead of SecurityType[167] (Refer to Volume 7 – Recommendations and Guidelines for Futures and Options Markets.") *- Part of 'Instrument' component block -* *
200	MaturityMonthYear	C	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified. *- Part of 'Instrument' component block -*
202	StrikePrice	C	Used for derivatives, such as options and covered warrants *- Part of 'Instrument' component block -*
210	MaxShow	C	
461	CFICode	C	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *- Part of 'Instrument' component block -* *

15	Currency	N	Identifies currency used for Price. Absence of this field is interpreted as the default for the security Must match original order.
37	OrderID	N	Unique identifier of most recent order as assigned by sell-side (broker, exchange, ECN).
58	Text	N	
59	TimeInForce	N	Absence of this field indicates Day order 0 = Day 1 = Good Till Cancel (GTC) 2 = At Opening 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK) 6 = Good To Date (GTD) S = Session *
77	PositionEffect	N	For use in derivatives omnibus accounting
100	ExDestination	N	Exchange code. Has to be agreed with both counter parties. You can find the list of available ExDestination on corresponding chapters below List of available ExDestination : XAMS XASX Australia Stock Exchange XBKK XBRU XCBT CBOT XCEC COMEX XCME CME XDMI XETR XEUR EUREX XHER XHKF Hong Kong Futures Exchange XHKG Stock Exchange Hong Kong XIDX XIPE ICE US XJAS XKFE Korean Futures Exchange XKLS XKOS Korea KOSDAQ XKRX Korea Exchange (Equities)

			XLIF LIFFE XMAT MATIF XMOO XNSE XNYF XNYM NYMEX XOSE Osaka Securities Exchange XPAR XPHS XSES Singapore SGX (Equities) XSFE Sydney Future Exchange XSIM Singapore SGX (Derivatives) XTAF Taiwan Fitires Exchange XTAI Taiwan Stock Exchange XTFF Tokyo TFX (aka TIFFE) XTKS Tokyo Stock Exchange XTKT Tokyo TOCOM
203	CoveredOrUncovered	N	For use with derivatives, such as options
465	QuantityType	N	
528	OrderCapacity	N	
529	OrderRestrictions	N	
582	CustOrderCapacity	N	
142	SenderLocationID (String)	Y	Mandatory for CME exchanges only, should be the country two characters. e.g. HK, JP
1028	ManualOrderIndicator (Char)	Y	Mandatory for CME exchanges only, indicates if the order is automatically generated (N) or manually (Y).
	Standard Trailer	Y	

* Details of these fields are given by Market further on the document

New Order – Multileg

The New Order - Multileg is provided to submit orders for securities that are made up of multiple securities, known as legs. Swaps, option strategies, futures spreads, are a few examples of multileg securities. A multileg security is made up of multiple securities that are traded atomically. This requirement that all legs be traded in the quantities that they make up the multlileg security is the important distinction between a multileg order and a list order.

Two generalized approaches to trading multileg securities are supported by FIX. The first approach involves a market maintaining multileg securities as separate products for which markets can be created. This "product approach" is often used in electronic trading systems. The second approach is to trade the multileg security as a group of separate securities – as is commonly done today in open outcry markets.

The multileg order can be traded using one of the following trading models using FIX. The first three models are variations on the multileg security as a separate tradeable product. The last models permits trading of multileg securities in environments where the multileg securities are not productized.

New Order - Multileg

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = AB
1	Account	Y	XXXNWGLLL
11	ClOrdID	Y	Unique identifier of the order as assigned by institution or by the intermediary with closest association with the investor. The rule to build the ClOrdID for Startegies isn't the same. You have to respect the following subpart XXX 3 characters to identify the client + Date (4 characters. YMDD Y last number of the year M Month in hexadecimal (1 = January, A = October, B = November & C = December) Increment number (5 characters) + 1 characters (R for a New Order, A for a Modification and C for a Cancel)
21	HandlInst	Y	Handling Instruction: 1 = automatic 2 = broker 3 = manual (care) Note: 2 behaves like automatic for the moment

40	OrdType	Y	1 = Market 2 = Limit 3 = Stop 4 = Stop Limit A = At Best O = Open *
54	Side	Y	Additional enumeration that indicates this is an order for a multileg order and that the sides are specified in the Instrument Leg component block.
60	TransactTime	Y	Time this order request was initiated/released by the trader, trading system, or intermediary. TimeStamp with following format: 20030620-10:06:51
555	NoLegs	Y	Number of legs Can be zero (e.g. standardized multileg instrument such as an Option strategy) – must be provided even if zero
600	LegSymbol	Y	Equivalent to the Symbol (55) for New Order - Single
22	SecurityIDSource	C	Required if SecurityID is specified. 1 = CUSIP Code 2 = SEDOL Code 3 = QUIK Code 4 = ISIN Code 5 = RIC Code - Part of 'Instrument' component block - *
55	Symbol	C	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists The value must correspond to an existing value in the destination system (see data dictionary) *- Part of 'Instrument' component block -* *
110	MinQty	C	Minimum quantity of an order to be executed
111	MaxFloor	C	Maximum number of shares within an order to be shown on the exchange floor at any given time
114	LocateReqd	C	Required for short sell orders

200	MaturityMonthYear	C	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified. *- Part of 'Instrument' component block -*
202	StrikePrice	C	Used for derivatives, such as options and covered warrants *- Part of 'Instrument' component block -*
210	MaxShow	C	
432	ExpireDate	C	Conditionally required if TimeInForce = GTD and ExpireTime is not specified.
461	CFICode	C	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *- Part of 'Instrument' component block -* *
15	Currency	N	Identifies currency used for Price. Absence of this field is interpreted as the default for the security
44	Price	N	Required for limit OrdTypes. For F/X orders, should be the "all-in" rate (spot rate adjusted for forward points). Can be used to specify a limit price for a pegged order, previously indicated, etc.
58	Text	N	
142	SenderLocationID (String)	Y	Mandatory for CME exchanges only, should be the country two characters. e.g. HK, JP
1028	ManualOrderIndicator (Char)	Y	Mandatory for CME exchanges only, indicates if the order is automatically generated (N) or manually (Y).
59	TimeInForce	N	Absence of this field indicates Day order 0 = Day 1 = Good Till Cancel (GTC) 2 = At Opening 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK) 6 = Good To Date (GTD) S = Session *
77	PositionEffect	N	For use in derivatives omnibus accounting
99	StopPx	N	Required for OrdType = "Stop" or OrdType = "Stop limit".

100	ExDestination	N	Exchange code. Has to be agreed with both counter parties. You can find the list of available ExDestination on corresponding chapters below List of available ExDestination : XAMS XASX Australia Stock Exchange XBKK XBRU XCBT CBOT XCEC COMEX XCME CME XDMI XETR XEUR EUREX XHER XHKF Hong Kong Futures Exchange XHKG Stock Exchange Hong Kong XIDX XIPE ICE US XJAS XKFE Korean Futures Exchange XKLS XKOS Korea KOSDAQ XKRX Korea Exchange (Equities) XLIF LIFFE XMAT MATIF XMOO XNSE XNYF XNYM NYMEX XOSE Osaka Securities Exchange XPAR XPHS XSES Singapore SGX (Equities) XSFE Sydney Future Exchange XSIM Singapore SGX (Derivatives) XTAF Taiwan Fitires Exchange XTAI Taiwan Stock Exchange XTFF Tokyo TFX (aka TIFFE) XTKS Tokyo Stock Exchange XTKT Tokyo TOCOM

126	ExpireTime		N	Conditionally required if TimeInForce = GTD and ExpireDate is not specified.
203	CoveredOrUncovered		N	For use with derivatives, such as options
423	PriceType		N	
528	OrderCapacity		N	
529	OrderRestrictions		N	
563	MultiLegRptTypeReq		N	Indicates the method of execution reporting requested by issuer of the order.
582	CustOrderCapacity		N	
608	LegCFICode		N	
610	LegMaturityMonthYear		N	
612	LegStrikePrice		N	
623	LegRatioQty		N	Specific to the <InstrumentLeg> (not in <Instrument>)
624	LegSide		N	Specific to the <InstrumentLeg> (not in <Instrument>)
→	*587*	*LegSettlmntTyp*	N	Refer to values for SettlmntTyp (63)
→	*654*	*LegRefID*	N	Used to identify a specific leg.
→	*566*	*LegPrice*	N	Provide only if a price is required for a specific leg. Used for anchoring the overall multileg security price to a specific leg price.
→	*564*	*LegPositionEffect*	N	Provide if the PositionEffect for the leg is different from that specified for the overall multileg security
→	*588*	*LegFutSettDate*	N	Refer to values for FutSettDate (64)
→	*565*	*LegCoveredOrUncovered*	N	Provide if the CoveredOrUncovered for the leg is different from that specified for the overall multileg security.
	Standard Trailer		Y	

* Details of these fields are given by Market further on the document

Multileg Order Cancel/Replace Request (a.k.a Multileg Order Modification Request)

Used to modify a multileg order previously submitted using the New Order - Multileg message. See Order Cancel Replace Request for details concerning message usage.

The format of the Multileg Order Cancel/Replace Request message is:

Multileg Order Cancel/Replace Request (a.k.a Multileg Order Modification Request)

Tag	Field Name	Req'd	Comments
	Standard Header	Y	MsgType = AC
37	OrderID	N	Unique identifier of most recent order as assigned by sell-side (broker, exchange, ECN).

41	OrigClOrdID	Y	ClOrdID of the previous order (NOT the initial order of the day) when canceling or replacing an order. The rule to build the ClOrdID for Strategies isn't the same. You have to respect the following subpart XXX 3 characters to identify the client + Date (4 characters. YMDD Y last number of the year M Month in hexadecimal (1 = January, A = October, B = November & C = December) Increment number (5 characters) + 1 characters (R for a New Order, A for a Modification and C for a Cancel)
50	SenderSubID	Y	This field is mandatory only for Order message and should not be populated for Admin messages (Logon, HB…) In case of Order messages, this field should be NEWEDGE_XXX XXX 3 characters to identify the client
11	ClOrdID	Y	Unique identifier of *replacement* order as assigned by institution or by the intermediary with closest association with the investor.. Note that this identifier will be used in ClOrdID field of the Cancel Reject message if the replacement request is rejected.
1	Account	N	XXXNWGLLL
21	HandlInst	Y	Handling Instruction: 1 = automatic 2 = broker 3 = manual (care) Note: 2 behaves like automatic for the moment
110	MinQty	C	Minimum quantity of an order to be executed
111	MaxFloor	N	Maximum number of shares within an order to be shown on the exchange floor at any given time
100	ExDestination	Y	Exchange code. Has to be agreed with both counter parties. You can find the list of available ExDestination on corresponding chapters below
54	Side	Y	Additional enumeration that indicates this is an order for a multileg order and that the sides are specified in the Instrument Leg component block.
555	NoLegs	Y	Number of legs Can be zero (e.g. standardized multileg instrument such as an Option strategy) – must be provided even if zero
600	LegSymbol	Y	Equivalent to the Symbol (55) for New Order -

				Single
608	LegCFICode		N	
610	LegMaturityMonthYear		N	
612	LegStrikePrice		N	
623	LegRatioQty		N	Specific to the <InstrumentLeg> (not in <Instrument>)
624	LegSide		N	Specific to the <InstrumentLeg> (not in <Instrument>)
→	564	LegPositionEffect	N	Provide if the PositionEffect for the leg is different from that specified for the overall multileg security
→	565	LegCoveredOrUncovered	N	Provide if the CoveredOrUncovered for the leg is different from that specified forthe overall multileg security.
→	654	LegRefID	N	Used to identify a specific leg.
→	566	LegPrice	N	Provide only if a price is required for a specific leg. Used for anchoring the overall multileg security price to a specific leg price.
→	587	LegSettlmntTyp	N	Refer to values for SettlmntTyp (63)
→	588	LegFutSettDate	N	Refer to values for FutSettDate (64)
114	LocateReqd		N	Required for short sell orders
60	TransactTime		Y	Time this order request was initiated/released by the trader, trading system, or intermediary.
465	QuantityType		N	
38	OrderQty		Y	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified. *- Part of 'Instrument' component block -*
40	OrdType		Y	1 = Market 2 = Limit 3 = Stop 4 = Stop Limit A = At Best O = Open *
44	Price		C	Required for limit OrdTypes. For F/X orders, should be the "all-in" rate (spot rate adjusted for forward points). Can be used to specify a limit price for a pegged order, previously indicated, etc.
99	StopPx		C	Required for OrdType = "Stop" or OrdType = "Stop limit".
15	Currency		N	Identifies currency used for Price. Absence of this field is interpreted as the default for the security

59	TimeInForce	N	Absence of this field indicates Day order 0 = Day 1 = Good Till Cancel (GTC) 2 = At Opening 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK) 6 = Good To Date (GTD) S = Session *
126	ExpireTime	C	Conditionally required if TimeInForce = GTD and ExpireDate is not specified.
528	OrderCapacity	N	
529	OrderRestrictions	N	
582	CustOrderCapacity	N	
58	Text	N	
77	PositionEffect	N	For use in derivatives omnibus accounting
203	CoveredOrUncovered	N	For use with derivatives, such as options
210	MaxShow	N	
563	MultiLegRptTypeReq	N	Indicates the method of execution reporting requested by issuer of the order.
	Standard Trailer	*Y*	

* Details of these fields are given by Market further on the document

COMMON COMPONENTS

Instrument (symbology) component block

<Instrument>				
Tag	Field Name		Req'd	Comments
55	Symbol		**Y**	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists (e.g. non-exchange traded Collective Investment Vehicles) *
65	SymbolSfx		N	
48	SecurityID		N	Takes precedence in identifying security to counterparty over SecurityAltID block. Requires SecurityIDSource if specified. *
22	SecurityIDSource		N	Required if SecurityID is specified. 1 = CUSIP Code 2 = SEDOL Code 3 = QUIK Code 4 = ISIN Code 5 = RIC Code *
454	NoSecurityAltID		N	Number of alternate Security Identifiers
	455	**SecurityAltID**	N	Security Alternate identifier for this security First member of repeating group - must be specified if NoSecurityAltID > 0 The Security Alternative identifier block should not be populated unless SecurityID and SecurityIDSource are populated and should not duplicate the SecurityID and SecurityIDSource values contained in the SecurityID/SecurityIDSource tags. Use of SecurityAltID may be used if bilaterally agreed to assist in security identification, and does not imply an obligation on the receiver of the message to ensure validity or consistency with the SecurityID and SecurityIDSource fields which take precedence.
	456	**SecurityAltID Source**	N	Source of SecurityAltID. Required if SecurityAltID is specified.
460	Product		N	Indicates the type of product the security is associated with (high-level category)

461	CFICode	N	Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values. It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. *
167	SecurityType	N	It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments. Futures and Options should be specified using the CFICode[461] field instead of SecurityType[167] (Refer to Volume 7 – Recommendations and Guidelines for Futures and Options Markets.") *
200	MaturityMonthYear	N	Specifies the month and year of maturity. Applicable for standardized derivatives which are typically only referenced by month and year (e.g. S&P futures). Note MaturityDate (a full date) can also be specified.
541	MaturityDate	N	Specifies date of maturity (a full date). Note that standardized derivatives which are typically only referenced by month and year (e.g. S&P futures).may use MaturityMonthYear and/or this field. When using MaturityMonthYear, it is recommended that markets and sell sides report the MaturityDate on all outbound messages as a means of data enrichment.
224	CouponPaymentDate	N	Date interest is to be paid. Used in identifying Corporate Bond issues.
225	IssueDate	N	Date instrument was issued. For Fixed Income IOIs for new issues, specifies the issue date.
239	RepoCollateralSecurityType	N	Identifies the collateral used in the transaction. For Fixed Income, required for RP and RVRP security types.
226	RepurchaseTerm	N	Number of business days before repurchase of a repo.
227	RepurchaseRate	N	Percent of par at which a Repo will be repaid. Represented as a percent, e.g. .9525 represents 95-1/4 percent of par.
228	Factor	N	Fraction for deriving Current face from Original face for TIPS, ABS or MBS Fixed Income securities. Note the fraction may be greater than, equal to or less than 1.
255	CreditRating	N	
543	InstrRegistry	N	The location at which records of ownership are maintained for this instrument, and at which ownership changes must be recorded. Can be used in conjunction with ISIN to address ISIN uniqueness issues.

470	CountryOfIssue	N	ISO Country code of instrument issue (e.g. the country portion typically used in ISIN). Can be used in conjunction with non-ISIN SecurityID (e.g. CUSIP for Municipal Bonds without ISIN) to provide uniqueness.
471	StateOrProvinceOfIssue	N	A two-character state or province abbreviation.
472	LocaleOfIssue	N	The three-character IATA code for a locale (e.g. airport code for Municipal Bonds).
240	RedemptionDate	N	Return of investor's principal in a security. Bond redemption can occur before maturity date.
202	StrikePrice	N	Used for derivatives, such as options and covered warrants
206	OptAttribute	N	Used for derivatives, such as options and covered warrants to indicate a versioning of the contract when required due to corporate actions to the underlying. Should not be used to indicate type of option – use the CFICode[461] for this purpose.
231	ContractMultiplier	N	For Fixed Income, Convertible Bonds, Derivatives, etc. Note: If used, quantities should be expressed in the "nominal" (e.g. contracts vs. shares) amount.
223	CouponRate	N	For Fixed Income.
207	SecurityExchange	N	Can be used to identify the security.
106	Issuer	N	
348	EncodedIssuerLen	N	Must be set if EncodedIssuer field is specified and must immediately precede it.
349	EncodedIssuer	N	Encoded (non-ASCII characters) representation of the Issuer field in the encoded format specified via the MessageEncoding field.
107	SecurityDesc	N	
350	EncodedSecurityDescLen	N	Must be set if EncodedSecurityDesc field is specified and must immediately precede it.
351	EncodedSecurityDesc	N	Encoded (non-ASCII characters) representation of the SecurityDesc field in the encoded format specified via the MessageEncoding field.
</Instrument>			

* Details of these fields are given by Market further on the document
* See Appendix B “Replaced Features and Supported Approach” and Appendix C “CFI Code”
Note: As advise on the FIX Protocol v4.3, we will replace the field 167 (SecurityType) by the field 461 CFICode.

Instrument for Strategies can be given by several fields. We advise to use the same Instrument code than the Single Leg Order. The Instrument should be populated:

On the common part of the New Order / Cancel/Replace – Multileg:
Symbol (Tag 55) + CFICode (Tag 461) (populated to FM for a Strategy Future or OM for a Strategy Option)
On the detailed part of each leg:
Symbol (Tag 55) + MaturityMonthYear (Tag 200) + CFICode (Tag 461) (populate to F for a Future Instrument – O for an Option)

Instrument Leg (symbology) component block

<InstrumentLeg>			
Tag	**Field Name**	**Req'd**	**Comments**
600	LegSymbol	Y	Equivalent to the Symbol (55) for New Order - Single
601	LegSymbolSfx	N	
602	LegSecurityID	N	
603	LegSecurityIDSource	N	
604	NoLegSecurityAltID	N	
	605 ***LegSecurityAltID***	N	
	606 ***LegSecurityAltIDSource***	N	
607	LegProduct	N	
608	LegCFICode	N	
609	LegSecurityType	N	
610	LegMaturityMonthYear	N	
611	LegMaturityDate	N	
248	LegCouponPaymentDate	N	
249	LegIssueDate	N	
250	LegRepoCollateralSecurityType	N	
251	LegRepurchaseTerm	N	
252	LegRepurchaseRate	N	
253	LegFactor	N	
257	LegCreditRating	N	
599	LegInstrRegistry	N	
596	LegCountryOfIssue	N	
597	LegStateOrProvinceOfIssue	N	

598	LeglocaleOfIssue	N	
254	LegRedemptionDate	N	
612	LegStrikePrice	N	
613	LegOptAttribute	N	
614	LegContractMultiplier	N	
615	LegCouponRate	N	
616	LegSecurityExchange	N	
617	LegIssuer	N	
618	EncodedLegIssuerLen	N	
619	EncodedLegIssuer	N	
620	LegSecurityDesc	N	
621	EncodedLegSecurityDescLen	N	
622	EncodedLegSecurityDesc	N	
623	LegRatioQty	N	Specific to the <InstrumentLeg> (not in <Instrument>)
624	LegSide	N	Specific to the <InstrumentLeg> (not in <Instrument>)
</InstrumentLeg>			

OrderQtyData component block

<OrderQtyData>			
Tag	**Field Name**	**Req'd**	**Comments**
38	OrderQty	N	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified.
152	CashOrderQty	N	One of CashOrderQty, OrderQty, or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified. Specifies the approximate “monetary quantity” for the order. Broker is responsible for converting and calculating OrderQty in tradeable units (e.g. shares) for subsequent messages.
516	OrderPercent	N	For CIV - Optional. One of CashOrderQty, OrderQty or (for CIV only) OrderPercent is required. Note that unless otherwise specified, only one of CashOrderQty, OrderQty, or OrderPercent should be specified.
468	RoundingDirection	N	For CIV – Optional
469	RoundingModulus	N	For CIV – Optional
</OrderQtyData>			

Market specifics

European Exchanges

Name	Description	Exchange code (tag 100)	Instruments	Order type				Order Time					
			Type	Market	Limit	Stop	Stop-Limit	Day	GTC	Market On Open	IOC	FOK	GTD
EUREX	German/Swiss Derivatives (Bund, DAX, Eurostoxx, SMI.)	EUREX	Futures, Options	Y	Y	Y		Y	Y		Y	Y	Y
Euronext AM	Amsterdam Stock Exchange	ENXT_AM	Equities	Y	Y	Y		Y	Y	Y	Y	Y	Y
AEX	Amsterdam Derivatives	AEX	Futures, Options	Y	Y			Y	Y				Y
Euronext Paris (ex MATIF)	Paris Derivatives (CAC40, index options, commodities…)	MATIF	Futures, Options	Y	Y	Y	Y	Y	Y		Y	Y	Y
Euronext Paris (ex MONEP)	Paris Equity Options (Options on French Stocks)	MONEP	Options	Y	Y			Y	Y		Y	Y	Y
Euronext VE	Paris Stock Exchange	ENXT_VE	Equities	Y	Y			Y	Y		Y	Y	Y
IDEM	Italian Derivatives (MIB…)	IDEM	Futures, Options	Y	Y		Y	Y	Y	Y	Y	Y	Y
LIFFE	UK Derivatives (Euribor, FTSE, Commodities) and Equity Options	LIFFE (GL=> LIFFEF & LIFFEE)	Futures, Options	Y	Y	Y	Y	Y	Y		Y	Y	Y
LSE (SETS)	London Stock Exchange	LSE	Equities	Y	Y			Y		Y	Y	Y	Y
MEFF	Spanish Derivatives (IBEX…)	MEFF	Futures, Options	Y	Y			Y	Y		Y	Y	
XETRA	German Stock Exchange	XETRA	Equities	Y	Y	Y	Y	Y	Y		Y	Y	Y
IPE	International Petroleum Exchange	IPE	Futures	Y	Y			Y		Y			
Euronext Brussels	Brussels Equities	ENXT_BR	Futures										

North-American Exchanges

Name	Description	Exchange code (tag 100)	Instruments	Order type				Order Time					
			Type	Market	Limit	Stop	Stop-Limit	Day	GTC	Market On Open	IOC	FOK	GTD
Globex (CME)	Eurodollars, S&P, Nasdaq, Currencies…	CME	Futures, Options	Y	Y	Y		Y	Y		Y	Y	Y
ECBOT	US Treasuries, Commodities, DJ Index…	CBOT	Futures, Options	Y	Y			Y	Y		Y	Y	Y
CBOTDIR	Floor trading: Wheat, Soybeans...	CBOTDIR	Futures	Y	Y	Y		Y	Y	Y			Y
Montreal Exchange	Derivatives	MX	Futures										
Montreal Options	Equity Options	MEOP		Y	Y			Y					
NYMEX	Commodities	NYMEX	Futures		Y			Y					
NYSE		NYSE	Equities	Y	Y			Y					
AMEX	Equities, Equity Options	AMEX, AMEXOP	Equities, Options	Y	Y			Y					
Chicago Stock Exchange		CHX	Equities	Y	Y			Y					
Island		ISLAND	Equities	Y	Y			Y					
ArcaEx		ARCA	Equities	Y	Y			Y					
Philadelphia Option Exchange	Equity Options	PHLX		Y	Y			Y					
Nasdaq Supersoes		SOES	Equities	Y	Y			Y					
Instinet		INSTINET		Y	Y			Y					
Knight		KNIGHT		Y	Y			Y					
Toronto Stock Exchange		TSES	Equities, Options	Y	Y			Y					
CBOE		CBOE		Y	Y			Y					
ISE	Equity Options	ISE		Y	Y			Y					
Boston Option Exchange	Equity Options	BOX		Y	Y			Y					
Pacific	Equity Options	PACIFIC		Y	Y			Y					
Screen2Pit	Screen on Trading Desk	S2P	Futures	Y	Y			Y					

Asian Exchanges

Name	Description	Exchange code (tag 100)	Instruments	Order type					Order Time						
			Type	Market	Limit	Stop	SLimit	FUNARI	Day	GTC	OPEN	CLOSE	IOC	FOK	GTD
SEHK	Hong Kong cash market	XHKG	Equities	Y	Y				Y				Y	Y	Y
KSE	Korean Stock Exchange	XKRX	Equities	Y	Y				Y						
TSE cash	Tokyo cash market	XTKS	Equities	Y	Y			Y	Y		Y	Y			
OSE cash	Osaka cash market	XOSE	Equities	Y	Y			Y	Y		Y	Y			
TSEC	Taiwan stock Exchange		Equities	Y	Y				Y						
SGX DT	Singapore Stock Exchange	XSES	Equities		Y				Y					Y	
ASX	Australian Stock Exchange	XASX	Equities		Y				Y						
JASDAQ	Jasdaq Securities Exchange		Equities	Y	Y				Y						
PSE	Philippines Stock Exchange		Equities		Y				Y						
IDX	Indonesia Stock Exchange		Equities		Y				Y						
KLSE	Kuala Lumpur Stock Exchange		Equities	Y	Y			Y	Y	Y	Y		Y		Y
SET	Thailand Stock Exchange		Equities	Y	Y				Y		Y	Y	Y	Y	
NSE	India – National Stock Exchange		Equities	Y	Y				Y				Y		
HKFE	Hong Kong Derivative	XHKF	Futures, options	Y	Y				Y		Y		Y	Y	
TSE	Tokyo Derivative market	XTKS	Futures, Options, JGB	Y	Y			Y	Y		Y	Y			
OSE	Osaka Derivative market	XOFE	Futures, Options	Y	Y			Y	Y		Y	Y			
TIFFE	Tokyo Financial Futures Exchange	XTXF	Futures	Y	Y				Y	Y			Y	Y	Y
TOCOM	Tokyo Commodity Exchange	XTKX	Futures	Y	Y		Y		Y	Y			Y	Y	Y
KRX	Korean Derivative market	XKFE	Futures, Options	Y	Y			Y	Y				Y	Y	
SFE	Sydney Futures Exchange	XKFE	Futures, options		Y				Y	Y			Y		Y
SGX	Singapore Exchange	XSIM	Futures, Options	Y	Y		Y		Y	Y			Y	Y	
TAIFEX	Taiwan Derivative market	XTAF	Futures, Options	Y	Y				Y				Y	Y	Y
NSE	India – National Stock Exchange	XNSE	Futures, Options	Y	Y				Y				Y		

APPENDIX – A – Multileg execution report

Background

A multileg security is made up of multiple securities that are traded automatically. Swaps, option strategies, futures spreads, are a few examples of multileg securities. This requirement that all legs be traded in the quantities that they make up the multileg security is the important distinction between a multileg order and a list order.

Model description

The FIX 4.3 protocol specification defines 4 models to support multileg orders. Ullink only supports the *SinGle Message Model* (Model4) for the moment.

FIX Client		**ULBridge**
Send Order on Strategy	> NEW ORDER MULTILEG>	
	< EXECUTION REPORT FOR MULTILEG < ExecType(150)=*New* Instrument Block = *Strategy Info* MultiLegReportingType(442)=*3* NoLegs(555)=n (*2 if spread)*	Send Market Acknowledgment
		…
	< EXECUTION REPORT FOR MULTILEG <	Strategies reported trade
	< EXECUTION REPORT FOR MULTILEG <	Leg 1 reported trade
	< EXECUTION REPORT FOR MULTILEG <	Leg n reported trade
		…

• Strategy legs description
• ExecType=*Trade*
• Instrument Block = *Strategy Info*
• MultiLegReportingType(442)=*3*
• OrdQty, LeavesQty, CumQty, AvgPx , LastQty, LastPrice apply to the overall strategy.
• NoLegs(555)=0
• ExecType=*Trade*
• Instrument Block = *Strategy Info*
• MultiLegReportingType(442)=*2*
• OrdQty, LeavesQty, CumQty, AvgPx , LastQty, LastPrice apply to the overall strategy.
• NoLegs(555)=*1*
  ○ Leg Description + LegRef
  ○ LeglastPrice is filled
  ○ Executed Qty for the leg can be calculated with : LegRatioQuantity*LastQty
• ExecType=*Trade*
• Instrument Block = *Strategy Info*
• MultiLegReportingType(442)=*2*
• OrdQty, LeavesQty, CumQty, AvgPx , LastQty, LastPrice apply to the overall strategy.
• NoLegs(555)=*1*
  ○ Leg Description + LegRef
  ○ LeglastPrice is filled
  ○ Executed Qty for the leg can be calculated with : LegRatioQuantity*LastQty

APPENDIX – B – Replaced Features and Supported Approach

Certain features of the FIX Protocol which were implemented in earlier versions of the FIX Protocol specification, have been removed and replaced by a different approach. Such features have been labeled as "Replaced" throughout the FIX Specification document. The replaced feature is no longer supported by this version of the FIX Protocol specification.

These features may or may not have been "Deprecated" in a previous version. Deprecation implies that implementations must support both approaches during the deprecation period. Removing and replacing a features without a deprecation period is based upon:

- Supporting both approaches would result in a high degree of complexity and/or ambiguity.

The rationale behind removing a feature is based upon either:

- Actual use and implementation of the feature identified major shortcomings necessitating a re-design.
- Additional business requirements have been identified which the feature is unable to expand and properly support in its present form.

The new, supported approach for each removed feature is identified below:

Replaced Field: ExecTransType (tag 20) and values in ExecType and OrdStatus fields [replaced in FIX 4.3]

The ExecTransType field introduced in FIX 2.7 has been removed and its values have been incorporated within the ExecType field. The ExecType field introducted in FIX 4.1 has had several values removed and a new value to represent the combination of the removed values. The ExecTransType and ExecType fields have effectively been merged into the ExecType field thus the removal of ExecTransType. Each field attempted to designate the type of Execution Report message received in a slightly different way. Both fields were designated as required. This became confusing and should be **simplified by a sinGle, merged field with the following mapping table:**

Removed Value within ExecTransType (20) field		Removed Value within OrdStatus (39) field		Removed Value within ExecType (150) field		ExecType (150)	
0	New						(various)
1	Cancel					H	Trade Cancel
2	Correct					G	Trade Correct
3	Status					H	Order Status
		5	Replaced			5	Replace
				1	Partial Fill	F	Trade
				2			Fill

2. Replaced Field Enumerations for Futures and Options for SecurityType (tag 167) with CFICode (tag 461) [replaced in FIX 4.3]

The CFICode was introduced to improve granularity in specifying security type. The adoption of CFICode has made the values for futures and options in SecurityType (tag 167) redundant.
The following Security Type values can be specified using CFICode via the following mapping table:

Value Removed From SecurityType (tag 167)		CFICode (tag 461) value
“FUT”	Future	First position of CFICode = “F”
“OPT”	Option	First position of CFICode = “O”

3. Replaced Field: PutOrCall (tag 201) and UnderlyingPutOrCall (tag 315) with CFICode (tag 461) [replaced in FIX 4.3]

The CFICode was introduced to improve granularity in specifying security type. The adoption of CFICode has made the PutOrCall (tag 201) redundant. The PutOrCall values are numeric and this has led to confusion on their usage as the data is not self describing. The CFICode uses a more readable format of “P” and “C” for put and call.

APPENDIX – C – CFI Code

CFICode Usage - ISO 10962 Classification of Financial Instruments (CFI code)

As of FIX 4.3, the CFICode field was added to the FIX Protocol in an attempt to provide a standards-based source of security type values by using values defined in ISO 10962 standard: Classification of Financial Instruments (CFI code). Prior to FIX 4.3, the SecurityType field was used to identify security types and was based upon a set of values published by ISITC (International Securities Association for Institutional Trade Communication) which ISITC no longer maintains.

It is recommended that CFICode be used instead of SecurityType for non-Fixed Income instruments as it is based upon an ISO standard and supports non-Fixed Income products in a manner consistent with the needs of FIX Protocol users. As of FIX 4.3, the SecurityType field was expanded and re-organized to support the requirements of Fixed Income products for FIX Protocol users, as the present ISO 10962 standard did not meet those needs.

ISO 10962 is maintained by ANNA (Association of National Numbering Agencies) acting as Registration Authority.
See the Product (460) and SecurityType (167) fields.
A subset of ISO 10962 values applicable to FIX usage are identified below. The official standard and set of possible values are maintained by the ISO 10962 standard and any discrepancies below should be considered typographical errors using the ISO 10962 standard as the correct set of values. To obtain the ISO 10962 standard, please contact the ISO 10962 secretariat and/or visit the ISO website at http://www.iso.ch

The ISO 10962 standard defines a 6 character code in which each character’s position value carries a special significance (attribute) and set of values. Note that "X" represents an unspecified or unknown attribute, thus it is not always necessary to specify every attribute (character position value).

High-level subset of possible values applicable to FIX usage:

Note: Corresponding FIX 4.2 SecurityType field value is identified within [ ]
ESXXXX = Equity Common Shares [CS]
EMXXXX = Equity Miscellaneous or Other (e.g. Exchange Traded Funds (ETFs), etc.) [n/a]
EPXXXX = Equity Preferred Shares [PS]
EUXXXX = Equity Units (unit trusts/mutual funds/OPCVM/OICVM) [MF]
DXXXXX = Debt (Fixed Income) [various]
DCXXXX = Debt Convertible Bond [CB]
FXXXXX = Future [FUT]
MRCXXX = Misc, Referential Instrument, Currency [FOR]
MRIXXX = Misc, Referential Instrument, Index [n/a]
MRRXXX = Misc, Referential Instrument, Interest Rate [n/a]

OCXXXX = Option - Call [OPT]
OPXXXX = Option - Put [OPT]
RWXXXX = Right Warrant [WAR]
XXXXXX = [NONE and ?]
Note that "X" represents an unspecified or unknown attribute and many of the values above containing "X" can be further subdefined according to the CFI standard (e.g. Voting rights are the third character of Equity Common Shares).

Detailed, granular subset of possible values applicable to FIX usage:

Options:
Definintion for Options (code defined by character position):

Char 1 *Category*	Char 2 *Group*	Char 3 *Scheme*	Char 4 *Underlying Asset*	Char 5 *Delivery*	Char 6
O=Options	C=Call P=Put M=Other X=Unknown(n/a)	A=American E=European X=Unknown(n/a)	B=Basket S=Stock-Equities D=Interest rate/notional debt sec T=Commodiites C=Currencies I=Indices O=Options F=Futures W=Swaps M=Other X=Unknown(n/a)	P=Physical C=Cash X=Unknown(n/a)	S=Standardized terms (maturity date, strike price, contract size) X=Unknown(n/a)

Examples:

OCXXXS	Standardized Call Option
OPXXXS	Standardized Put Option
OCXFXS	Standardized Call Option on a Future
OPXFXS	Standardized Put Option on a Future
OCEFCN	Nonstandard (flex) call option on future with european style expiration and cash delivery
OPAFPN	Nonstandard (flex) put option on future with american style expiration and physical delivery
OPXSPN	Nonstandard (flex) put option on a stock with physical delivery (the expiration style is not specified – so is assumed to default to the market standard for flex options).
OCEICN	Nonstandard (flex) call option on an index with european style expiration and cash delivery

Futures:

Definition for Futures (code defined by character position):

Char 1 *Category*	Char 2	*Underlying Asset*	Char 4 *Delivery*	Char 5 *Stdized/Non-Std*	Char 6 *N/A Undefined*
F=Futures	F=Financial Futures C=Commodity Futures M=Others X=Unknown(n/a)	A=Agriculture, forestry, and fishing B=Basket S=Stock-Equities (for financial future) or Services (for commodities futures) D=Interest rate/notional debt sec C=Currencies I=Indices (for financial futures) or Industrial Products (for commodities futures) O=Options F=Futures W=Swaps M=Other X=Unknown(n/a)	P=Physical C=Cash	S=Standardized terms (maturity date, strike price, contract size) N=Non-standardized terms X=Unknown(n/a)	X=Not applicable / undefined

Examples:

FXXXS	Standardized Future
FFICN	Nonstandard (flex) Financial Future on an index with cash delivery
FCEPN	Nonstandard (flex) Commodity Future on an extraction resource with physical delivery
FXXXN	Nonstandard (flex) future – contract type specified in symbology – not provided in CFICode