FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX)

Version 5.0

VOLUME 1 – INTRODUCTION TO THE FIX PROTOCOL

December 2006

DISCLAIMER

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

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

No proprietary or ownership interest of any kind is granted with respect to the FIX Protocol (or any rights therein), except as expressly set out in FIX Protocol Limited's Copyright and Acceptable Use Policy..

REPRODUCTION

FIX Protocol Limited grants permission to print in hard copy form or reproduce the FIX Protocol specification in its entirety provided that the duplicated pages retain the “Copyright FIX Protocol Limited” statement at the bottom of the page.

Portions of the FIX Protocol specification may be extracted or cited in other documents (such as a document which describes one’s implementation of the FIX Protocol) provided that one reference the origin of the FIX Protocol specification (***http://www.fixprotocol.org***) and that the specification itself is “Copyright FIX Protocol Limited”.

FIX Protocol Limited claims no intellectual property over one’s implementation (programming code) of an application which implements the behavior and details from the FIX Protocol specification.

PREFACE

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

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

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

FIX is now used by a variety of firms and vendors. It has clearly emerged as the inter-firm messaging protocol of choice. FIX has grown from its original buyside-to-sellside equity trading roots. It is now used by markets (exchanges, "ECNs", etc) and other market participants. In addition to equities, FIX currently supports four other products: Collective Investment Vehicles (CIVs), Derivatives, Fixed Income, and Foreign Exchange. The process for modifications to the specification is very open with input and feedback encouraged from the community. Those interested in providing input to the protocol are encouraged use the FIX website Discussion section or contact the FIX Global Technical Committee Chairpersons, Kevin Houstoun, HSBC, or Matt Simpson, Chicago Mercantile Exchange. **The FIX website at *http://www.fixprotocol.org* is the main source of information, discussion, and notification of FIX-related events.**

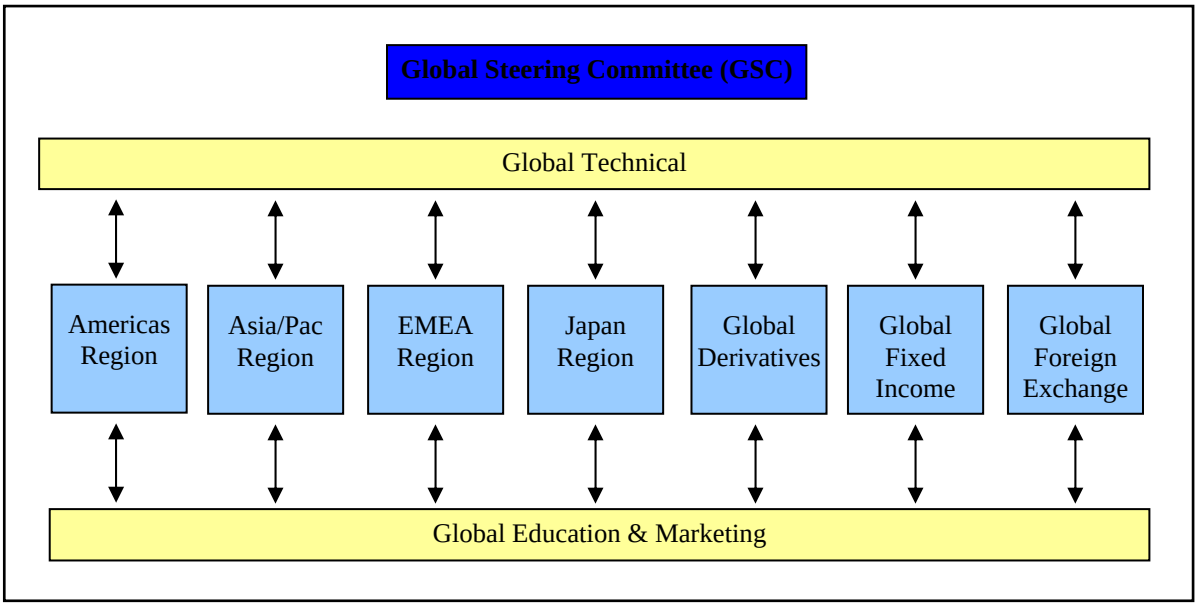
We look forward to your participation.

FIX Protocol Ltd December 2006

About FIX Protocol Limited

FIX Protocol Limited (FPL) (***http://www.fixprotocol.org***) oversees and manages the development of the FIX Protocol specification and encourages its use throughout the industry. FPL is open to due paying members representing business and technology professionals interested in guiding the growth and adoption of the FIX Protocol that work for: Buy-side Firms, Sell-side Firms, Exchanges, ECNs/ATSs, Utilities, Vendors, and Other Associations. For more information about membership please visit ***http://www.fixprotocol.org/join/*** .

FIX Protocol Limited is represented by the following high-level organization structure:



- Global Steering Committee comprised of the FPL Committee Chairs
- Global Technical and Global Education & Marketing comprised of Product/Region Committee Representatives

For a current list of FPL Member firms, visit:

http://www.fixprotocol.org/members/

For a current list of active FPL Working Groups, visit:

http://www.fixprotocol.org/working_groups/

Links to Product and Regional Committees' web pages are at:

http://www.fixprotocol.org/committees/

VOLUME INDEX

VOLUME 5 - FIX APPLICATION MESSAGES: POST-TRADE

VOLUME 6 - FIX DATA DICTIONARY

VOLUME 7 - FIX USAGE BY PRODUCT

Contents – Volume 1

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 cross-border trading, derivatives, fixed income, and other products. 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.

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 divided into volumes and organized to reflect the distinction.

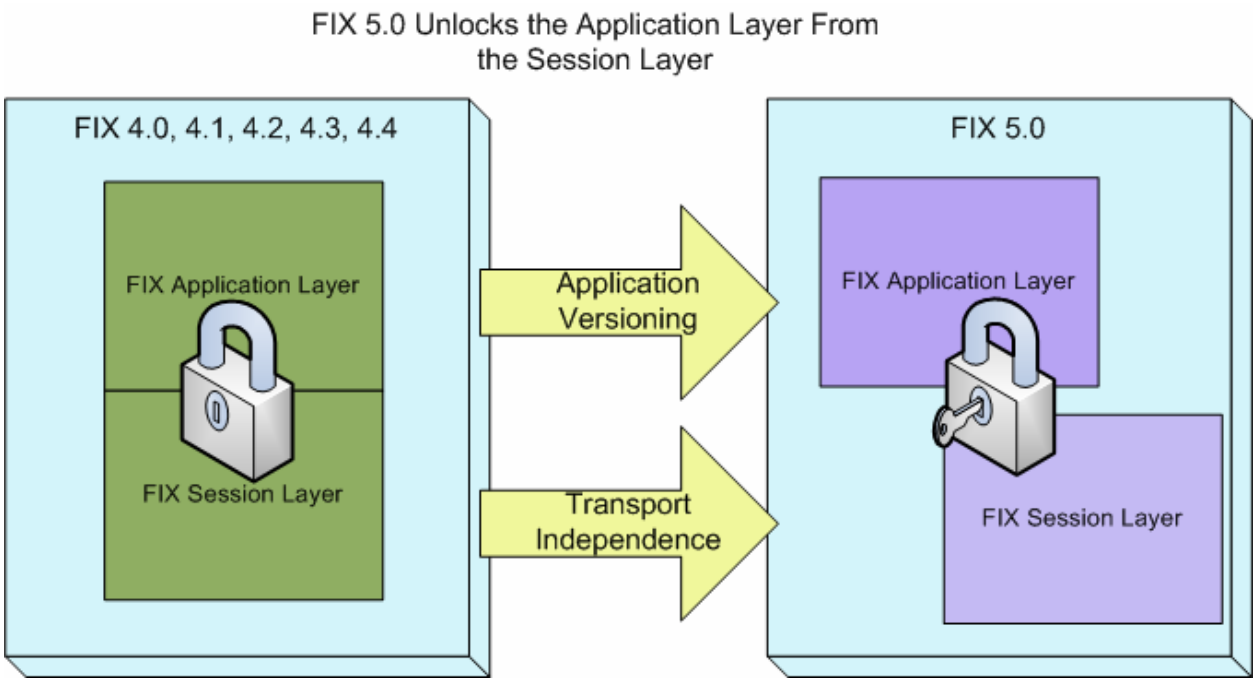
DOCUMENT NAVIGATION

One useful tip when navigating within a volume is to take advantage of the fact that each document contains "bookmarks" to its main sections. You can use the word processor's "Goto" function (i.e. Ctrl-G) to quickly navigate from one key section or appendix to another.

Third parties or volunteers have historically built useful utilities "generated" using the specification document as their basis which provide cross-reference and lookup capabilities. Such free utilities are available via the FIX website.

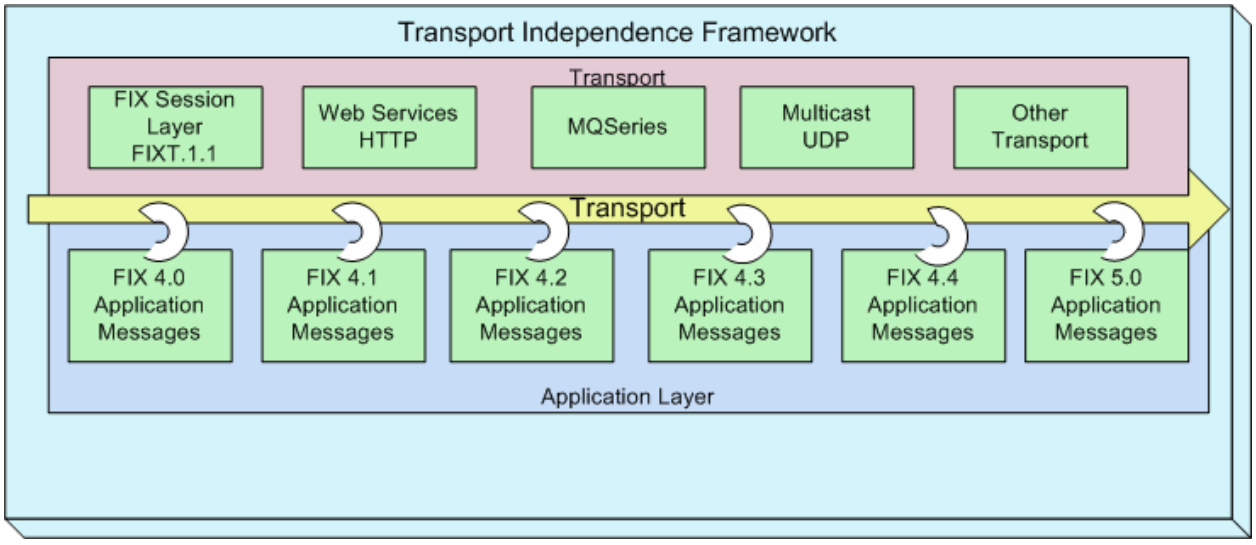
OVERVIEW OF MAJOR CHANGES IN FIX 5.0

With the release of FIX 5.0 in October 2006, the FPL Global Technical Committee (GTC) introduced a new framework, the transport independence (TI) framework, which separated the FIX Session Protocol from the FIX Application Protocol. Under the TI framework the application protocol messages can be sent over any suitable session transport technology (e.g. WS-RX, MQ, publish/subscribe message bus), where the FIX Session Protocol is one of the available options as a session transport for FIX application messages. From this release forward the FIX Application layer and the FIX Session layer will have their own versioning moniker. The FIX Application layer will retain the traditional version moniker of "FIX x.y" while the FIX Session layer will utilize a new version moniker of "FIXT x.y" (note that the version numbers will be independent of each other). The diagram below illustrates how previously the FIX Session layer was tighly coupled to the Application layer. With the advent of Application Versioning and Transport Independence, the FIX Session and Application layers have been decoupled and are now independent.



Transport Independence (TI) Framework

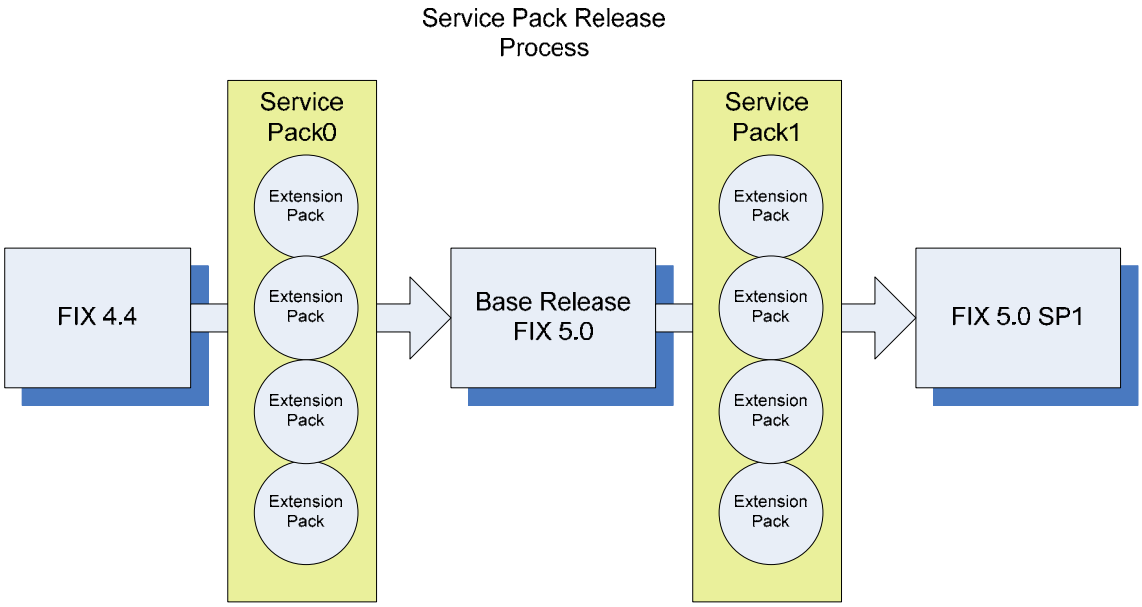
The transport independence (TI) framework separates the previously coupled FIX Session layer from the FIX Application layer. Under this framework the FIX Application Protocol can use any transport technology in addition to the FIX Session Protocol. The diagram below illustrates how various transport mechanisms, including the FIX Session layer, can be used to carry the full suite of FIX Application versions.



To support this framework a key new field has been added called *ApplVerID* (application version ID, tag 1128). Depending on the use case *ApplVerID* may be optional or required. Additionally, the FIX field *BeginString* will no longer identify the FIX application version, but identifies the FIX Session Protocol version. The sections below discusses the four main uses cases supported by the TI framework.

Application Versioning

Application Versioning allows extensions to the current base application version to be applied using a formal release process. Extension Packs represent the individual gap analysis proposals submitted to the GTC for review and approval. Extension Packs are grouped into Service Packs and are applied to the base application version, usually the most current FIX application version. A new application version is formed when a new Service Pack is applied to a base version. In the diagram below, FIX 4.4 has been extended via Service Pack 0, forming a new application version called FIX 5.0. As new Extension Packs are approved they will be grouped into Service Pack 1 which is then released to form the next application version identified as FIX 5.0 SP1. These application versions are expressed using the new tag *ApplVerID*.



Service Pack Mangement

ApplVerID is an enumerated field. These enumerations are used to express prior versions of FIX inclusive of FIX 4.0, 4.1, 4.2, 4.3 and 4.4 as well as the most recent version, FIX 5.0. Going forward, service packs will be applied to the base version, in this case FIX 5.0, and will be identified as FIX Version + Service Pack . This means that FIX 5.0 will be represented as an enumeration (7) rather than as an actual value in the ApplVerID field. Service Pack identifiers will consist of the base FIX version, the service pack number for that version, and the date the service pack was released. For example, the assigned value for service pack 1 may be “FIX 5.0 SP1 June 30, 2007”.

Extension Pack Mangement

Extension Packs are the building blocks of a Service Pack and represent specific functional proposals that have been presented to the GTC. Prior to the release of a Service Pack, Extension Packs are applied to the most recent version of the repository so that they can be used at the point they become available. Extension Packs are applied to the repository in a cumulative manner and will at some point culminate in a Service Pack release. Extension Packs management will be conducted as follows:

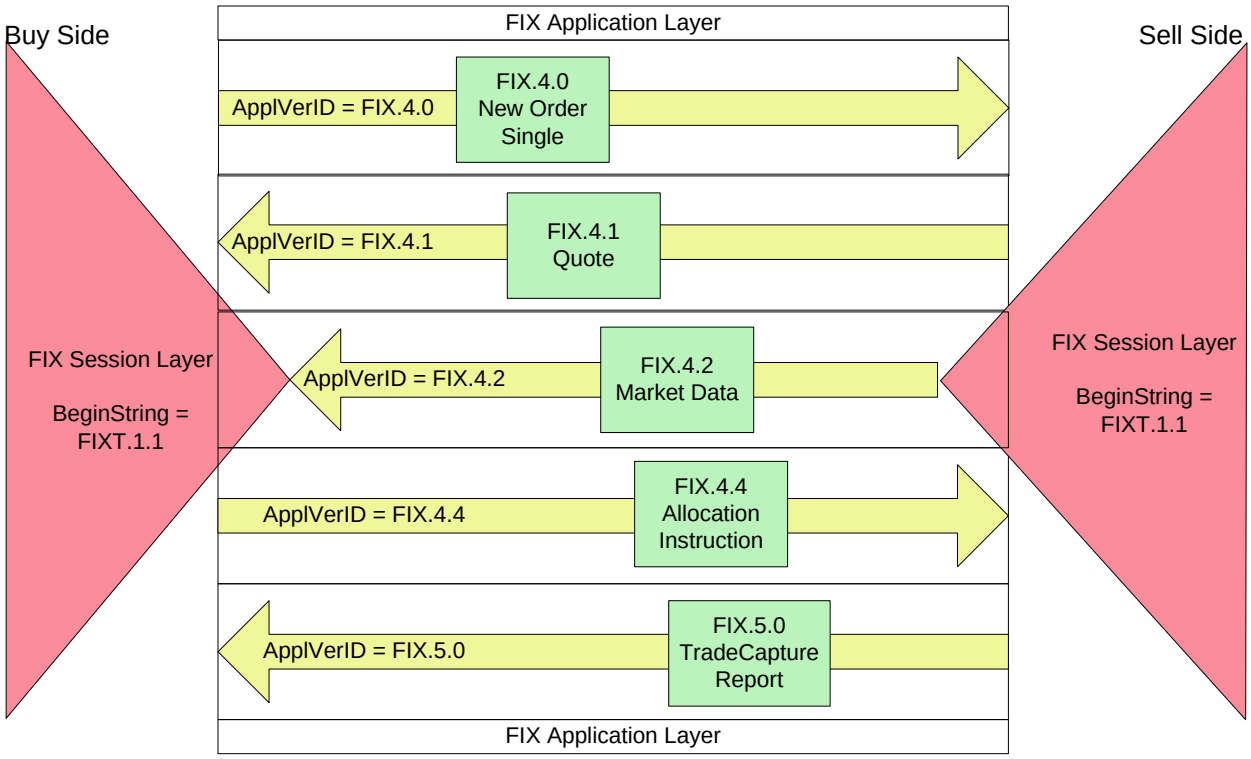
1. Extension Packs will be assigned a unique, sequential number at the point they are approved by the GTC
2. Extension Packs are applied to the most recent version of the repository and may be inclusive of prior Extension Packs
3. At the pont an Extension Pack has been applied, the updated repository, schema, and message tables will be available

4. When implementing a specific Extension Pack, the field CustomApplVerID (1129) will be used to specify the Extension Pack Identifier
5. User's of an Extension Pack need not implement other Extension Packs present in the repository. Rules of engagement need to be bilaterally agreed on.

Flexibility Provided by FIX 5.0

This is the 'GTC approved' approach which separates the FIX session layer from the application layer, provides support for application versioning, and creates a platform for transport independence. This approach will treat the FIX session like 'any other' transport and allow the unambiguous use of any application version via the *ApplVerID* field. A value of FIXT.1.1 in the *BeginString* of the FIX Session will indicate that application versioning is in effect and the version should be determined either through the Logon's *NoMsgType* repeating group or the *AppVerID* field. Future extensions to the session layer or application layer will be supported independent of each other as point releases to *BeginString* and *ApplVerID*, respectively. Major Tags describing the session and application versions are: BeginString=FIXT.1.1 (or later versions) and ApplVerID=FIX.5.0[1] (or later versions). A BeginString=FIX.5.0 (or later versions) will not be valid.

The diagram below illustrates how the new FIXT.1.1 Session layer can be used to transport makes use of the ApplVerID in the Application layer in order to support a broad set of application versions. [2]



[1] The value FIX.5.0 will be represented using enumeration 7

[2] FIX.4.0, FIX.4.1, FIX.4.2, FIX.4.3, FIX.4.4, FIX.5.0 are represented using enumerations

FIX PROTOCOL SYNTAX

The FIX Protocol currently exists in two syntaxes:

1. "Tag=Value" syntax
2. FIXML syntax

The same business message flow applies to either syntax. A specific syntax is simply a slightly different way to represent the same thing in much the same way that "3" and "three" represent the same thing.

COMMON FIX SYNTAX RULES

The following section summarizes general specifications for constructing FIX messages which are applicable to both "Tag=Value" and FIXML syntaxes.

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.
  - **DayOfMonth:** int field (see definition of "int" above) representing a day during a particular month (values 1-31).

- **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" = "23.").

  Note that fields which are derived from float may contain negative values unless explicitly specified otherwise.

- **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. For certain asset classes prices may be negative values. For example, prices for options strategies can be negative under certain market conditions. Refer to ***Volume 7: FIX Usage by Product*** for asset classes that support negative price values.
- **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. 0.05 represents 5% and 0.9525 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**).
  - **MultipleCharValue:** String field (see definition of “String” above) containing one or more **single character** space delimited values.
  - **MultipleStringValue:** String field (see definition of "String" above) containing one or more **multiple character** space deliminted values.
  - **Country:** String field (see definition of “String” above) representing a country using ISO 3166 Country code (2 character) values.

    Valid values:

    See "Appendix 6-B - FIX Fields Based Upon Other Standards"
  - **Currency:** String field (see definition of “String” above) representing a currency type using ISO 4217 Currency code (3 character) values.

    Valid values:

    See "Appendix 6-A - Currency Codes - ISO 4217 Currency codes"
  - **Exchange:** String field (see definition of “String” above) representing a market or exchange.

    Valid values:

    See "Appendix 6-C - Exchange Codes - ISO 10383 Market Identifier Code (MIC)"

- **month-year:** String field representing month of a year. An optional day of the month can be appended or an optional week code.

  Valid formats:

  YYYYMM

  YYYYMMDD

  YYYYMMWW

  Valid values:

  YYYY = 0000-9999, MM = 01-12, DD = 01-31, WW = w1, w2, w3, w4, w5.

- **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. This special-purpose field is paired with UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages.

  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).

- **UTCDateOnly:** Date represented in UTC (Universal Time Coordinated, also known as "GMT") in YYYYMMDD format. This special-purpose field is paired with UTCTimeOnly to form a proper UTCTimestamp for bandwidth-sensitive messages.

  Valid values:

  - YYYY = 0000-9999, MM = 01-12, DD = 01-31.

- **LocalMktDate:** Date of Local Market (vs. UTC) in YYYYMMDD format. This is the "normal" date field used by the FIX protocol.

  Valid values:

  - YYYY = 0000-9999, MM = 01-12, DD = 01-31.

- **TZTimeOnly:** The time represented based on ISO 8601. This is the time with a UTC offset to allow identification of local time and timezone of that time.

Format is HH:MM[:SS][.sss][Z | [+ | - hh[:mm]]] where HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, sss = milliseconds, hh = 01-12 offset hours, mm = 00-59 offset minutes.

Example: 07:39Z is 07:39 UTC

Example: 02:39-05 is five hours behind UTC, thus Eastern Time

Example: 15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore time

Example: 13:09+05:30 is 5.5 hours ahead of UTC, India time

- **TZTimestamp:** (based on existing FIX data type of String). The time/date combination representing local time with an offset to UTC to allow identification of local time and timezone offset of that time. The representation is based on ISO 8601.

  Format is YYYYMMDD-HH:MM:SS[.sss][Z | [+ | - hh[:mm]]] where

  YYYY = 0000 to 9999, MM = 01-12, DD = 01-31

  HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, sss = milliseconds, hh = 01-12 offset hours, mm = 00-59 offset minutes

  Example: 20060901-07:39Z is 07:39 UTC on 1st of September 2006

  Example: 20060901-02:39-05 is five hours behind UTC, thus Eastern Time on 1st of September 2006

  Example: 20060901-15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore time on 1st of September 2006

  Example: 20060901-13:09+05:30 is 5.5 hours ahead of UTC, India time on 1st of September 2006

- **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".*

- **Pattern Data Type**

  A "pattern" data type is used to build on and provide some restrictions on what is allowed as valid values in fields that uses a base FIX data type and a pattern data type. The universe of allowable valid values for the field would then be the union of the base set of valid values and what is defined by the pattern data type. The pattern data type used by the field will retain its base FIX data type (e.g. String, int, char).

  - **Tenor:** the Tenor pattern is used to allow the expression of FX standard tenors in addition to the base valid enumerations defined for the field that uses this pattern data type. This pattern data type is defined as follows:

    Dx = FX tenor expression for "days", e.g. "D5" to express 5 days, where "x" is any integer > 0

    Mx = FX tenor expression for "months", e.g. "M3" to express 3 months, where "x" is any integer > 0

    Wx = FX tenor expression for "weeks", e.g. "W13" to express 13 weeks, where "x" is any integer > 0

    Yx = FX tenor expression for "years", e.g. "Y1" to express 1 year, where "x" is any integer > 0

  - **Reserved100Plus:** the Reserved100Plus pattern is used to allow additional billaterally agreed upon enumerations to be defined for the field by using enumeration values starting at "100" and above.

- **Reserved1000Plus:** the Reserved1000Plus pattern is used to allow additional billaterally agreed upon enumerations to be defined for the field by using enumeration values starting at "1000" and above.
- **Reserved4000Plus:** the Reserved4000Plus pattern is used to allow additional billaterally agreed upon enumerations to be defined for the field by using enumeration values starting at "4000" and above.

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.

FIX "Tag=Value" SYNTAX

The following section summarizes general specifications for constructing FIX messages in "Tag=Value" syntax.

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.**
5. **A tag number (field) should only appear in a message once. If it appears more than once in the message it should be considered an error with the specification document. The error should be pointed out to the FIX Global Technical Committee.**

In addition, certain fields of the data type *MultipleCharValue* 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'). Fields of the data type *MultipleStringValue* can contain multiple values that consists of string values separated by a space within the "value" portion of that field followed by a single "SOH" character (e.g. "277=AA I AJ<SOH>" represents 3 values: 'AA', 'I', 'AJ').

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 e.g.* 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 first field listed after the NoXXX, then becomes conditionally required if the NoXXX field is greater than zero.
- The NoXXX field (for example: NoTradingSessions, NoAllocs) which specifies the number of repeating group instances occurs once for a repeating group and must immediately precede the repeating group contents.
- The NoXXX field is required if one of the fields in the repeating group is required. If all members of a repeating group are optional, then the NoXXX field should also be optional.
- 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

Example of a repeating group:

Part of message				
215	NoRoutingIDs		N	Required if any RoutingType and RoutingIDs are specified. Indicates the number within repeating group.
→	***216***	***RoutingType***	N	Indicates type of RoutingID. Required if NoRoutingIDs is > 0.
→	***217***	***RoutingID***	N	Identifies routing destination. Required if NoRoutingIDs is > 0.
Rest of the message not shown				

Example of nested repeating group

Portion of New Order - List message showing a nested repeating group for allocations for each order. Note the NoAllocs repeating group is nested within the NoOrders repeating group and as such each instance of the orders repeating group may contain a repeating group of allocations.					
73	NoOrders			Y	Number of orders in this message (number of repeating groups to follow)
→	***11***	***ClOrdID***		Y	Must be the first field in the repeating group.
→	***526***	***SecondaryClOrdID***		N	
→	***67***	***ListSeqNo***		Y	Order number within the list
→	***583***	***ClOrdLinkID***		N	
→	***160***	***SettlInstMode***		N	
→	***component block <Parties>***			N	Insert here the set of "Parties" (firm identification) fields defined in "COMMON COMPONENTS OF APPLICATION MESSAGES"
→	***229***	***TradeOriginationDate***		N	
→	***1***	***Account***		N	
→	***581***	***AccountType***		N	
→	***589***	***DayBookingInst***		N	
→	***590***	***BookingUnit***		N	
→	***591***	***PreallocMethod***		N	
→	***78***	***NoAllocs***		N	Indicates number of pre-trade allocation accounts to follow
→	→	***79***	***AllocAccount***	N	Required if NoAllocs > 0. Must be the first field in the repeating group.
→	→	***467***	***IndividualAllocID***	N	
→	→	***component block <NestedParties>***		N	Insert here the set of "Nested Parties" (firm identification "nested" within additional repeating group) fields defined in "COMMON COMPONENTS OF APPLICATION MESSAGES"
→	→	***80***	***AllocQty***	N	
→	***63***	***SettlmntTyp***		N	
→	***64***	***FutSettDate***		N	Takes precedence over SettlmntTyp value and conditionally required/omitted for specific SettlmntTyp values.
Rest of the message not shown					

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 Global 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 communcation. 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.

Example Usage of Encoded Fields For non-ASCII Language Support

The examples below illustrates how the MessageEncoding (347) field is used in conjunction with the various available encoded fields in FIX.

Example 1 - Specify the ASCII/English value as Issuer plus Japanese character set as EncodedIssuer

Tag	Field Name	Value
…Other Standard Header fields		
347	MessageEncoding	Shift_JIS
…Other Standard Header fields		
…Other Message Body fields		
106	Issuer	HITACHI
348	EncodedIssuerLen	10
349	EncodedIssuer	日立製作所
…Other Message Body fields		

Example 2 - Specify the ASCII/English value as Issuer plus Japanese character set as EncodedIssuer. Specify the ASCII/English value as Text plus Japanese character set as EncodedText.

Tag	Field Name	Value
…Other Standard Header fields		
347	MessageEncoding	Shift_JIS
…Other Standard Header fields		
…Other Message Body fields		
106	Issuer	HITACHI
348	EncodedIssuerLen	10
349	EncodedIssuer	日立製作所
…Other Message Body fields		
58	Text	This is a test
356	EncodedTextLen	17
357	EncodedText	これはテストです。
…Other Message Body fields		

Precautions when using UNICODE

There is the possibility that an SOH may be included in the character data when using UNICODE encoding. To avoid parsing problems, a FIX engine should use the EncodedLen value to extract the proper number of bytes.

FIXML SYNTAX

FIXML Highlights

- FIXML is the XML vocabulary for creating FIX messages.
- Uses the same FIX data dictionary and business logic.
- Focuses primarily on the FIX Application Messages and does not provide a session layer.
- Can be encapsulated within the FIX Session Protocol or within another protocol like, MQ Series, TIBCO, SOAP, etc.

Background

The FPL FIXML Working Group began investigating the XML format in 1998 and published a White Paper supporting an evolutionary approach to migrating the FIX Protocol to an XML format. The working group released an intial version of the FIXML DTDs on January 15th, 1999. There are currently DTDs based on FIX Protocol versions 4.1, 4.2 and 4.3. A FIXML Schema based version of FIXML was released following the release of FIX 4.4.

The FIXML language is in a state of transition. It has been four years since the initial release of FIXML. XML technology has advanced considerably in those four years. FPL committed to deliver an XML Schema representation for FIXML starting with FIX 4.3. Issues confronting FIXML users in the derivatives post trade area preempted release of the FIXML Schema for FIX 4.3. Instead the effort shifted to attempts to exploit the capabilities available in XML Schema to define a version of FIXML that was optimized to reduce message size. This version of FIXML was referred to as Transport Optimized FIXML during its development. The Global Technical Committee chose to release the transport optimizations in two phases.

The **FIX 4.4 DTD Version** was released with FIX 4.4 Eintroduced standardized abbreviations for field names and removal of container elements used to represent repeating groups and component blocks. **This version has been replaced by the FIX 4.4 Schema Version and should no longer be used.**

The **FIX 4.4 Schema Version** was released as part of FIX 4.4 Errata release. The FIX 4.4 Schema Version exploits the enhanced capabilities of XML Schema to further optimize FIXML message size by introducing the use of attributes to represent fields.

FIXML for FIX 5.0 is defined by an XML Schema based upon the work done for FIX 4.4.

FIX and FIXML Version and Comparison using New Order Single Message

The following section compares the implementation of the same FIX new order single message in FIX 4.2 tag=value format, FIXML 4.2 DTD version, and FIXML Schema Version.

FIX tag=value Version

The following is a FIX 4.2 New Order Single message in classic tag-value pair format:

```
8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2^52=20030615-
01:14:49^11=12345^1=111111^63=0^64=20030621^21=3^110=1000^111=50000^55=IBM^48=4592001
01^22=1^54=1^60=2003061501:14:49 38=5000^40=1^44=15.75^15=USD^59=0^10=127
```

NOTE: ^ represents the SOH separator.

The message is 195 bytes in length.

FIXML 4.2 Version

The following is a roughly equivalent FIXML 4.2 DTD-based message:

```
<FIXML>
      <FIXMLMessage>
            <Header>
                  <PossDupFlag Value="N" />
                  <PossResend Value="N" />
                  <SendingTime>20020103-12:00:01</SendingTime>
                  <Sender>
                        <CompID>AFUNDMGR</CompID>
                  </Sender>
                  <Target>
                        <CompID>ABROKER</CompID>
                  </Target>
            </Header>
            <ApplicationMessage>
                  <Order>
                        <ClOrdID>1968</ClOrdID>
                        <Account>4130287</Account>
                        <HandlInst Value="1" />
                        <ExDestination Value="L" />
                        <Instrument>
                              <Symbol>IBM</Symbol>
                              <SecurityID>459200101</SecurityID>
                              <SecurityIDSource Value="1" />
                        </Instrument>
                        <Side Value="2" />
                        <TransactTime>20021120-12:13:12</TransactTime>
                        <OrderQtyData>
                              <OrderQty>1000</OrderQty>
                        </OrderQtyData>
                        <OrdType Value="2" />
                        <Price>93.25</Price>
                        <Currency Value="USD" />
                  </Order>
            </ApplicationMessage>
      </FIXMLMessage>
</FIXML>
```

This message is 684 bytes; over three times the message size of the raw FIX tav=value message. In practice, FIXML messages could be 3-5 times their FIX tag=value equivalents.

FIXML 4.4 Schema Version

The following is a New Order Single message based on the FIXML 4.4 Schema.

```
<FIXML>
     <Order ClOrdID="123456"
                    Side="2"
                    TransactTm="2001-09-11T09:30:47-05:00"
                    OrdTyp="2"
                    Px="93.25"
                    Acct="26522154">
          <Hdr Snt="2001-09-11T09:30:47-05:00"
                  PosDup="N"
                  PosRsnd="N"
                  SeqNum="521">
           <Sndr ID="AFUNDMGR"/>
```



```
            <Tgt ID="ABROKER"/>
          </Hdr>
          <Instrmt Sym="IBM"
                     ID="459200101"
                     IDSrc="1"/>
          <OrdQty Qty="1000"/>
        </Order>
</FIXML>
```

NOTE: The XML attributes in the message have been placed on separate lines to aid readability

This message is 348 bytes in length; approximately 70% larger than the raw FIX tag=value message, but roughly half the size of the previous FIXML format without significant loss in readability.

Sample Message Content

The following table is included to help clarify the message content shown above

Tag/Attribute	Meaning
<FIXML>	Root element
<Order ClOrdID="123456" Side="2" TransactTm="2001-09-11T09:30:47-05:00" OrdTyp="2" Px="93.25" Acct="26522154">	New order Client's order ID Sell order Transaction time Limit order Limit price Customer's account
<Instrmt Sym="IBM" ID="459200101" IDSrc="1"/>	Stock symbol Stock CUSIP (ID source=CUSIP)
<OrdQty Qty="1000"/>	Order quantity
</Order>	Close of order
</FIXML>	Close root element

FIXML Transition to Schema

FIXML was initiated at a time when the only mechanism available to define and validate an XML syntax was the Document Type Definition (DTD) originally created as part of the Standardized General Markup Language (SGML). The DTD provided only minimal ability to define XML syntax.

Since then, the World Wide Web Consortium (http://www.w3c.org) adopted XML Schema as a way of representing the format of XML messages using XML syntax. Document Type Definitions (DTDs), which were originally part of XML, have limited syntax and capabilities for defining XML syntax. XML Schema was designed to address many of the deficiencies of DTDs. The FPL Global Technical Committee has received numerous requests from FIX users for an XML Schema representation of the FIX Protocol and believes that a version of FIXML defined using XML Schema will provide a more robust, optimized message format and provide a better environment for users implementing FIXML applications.

The following limitations of DTDs determined much of the FIXML implementation;

Meta data could not be included in the DTD - so attributes were used for meta-data.

Attributes could not be "typed" so this restricted datatyping to elements. Many XML syntax's then relied heavily on elements for data, attributes for meta-data. This is the approach taken for FIXML up through the FIX 4.4 Errata 20030618 release.

Since the initial release of FIXML in 1999, XML technology has advanced. The primary advancement has been in the area of standards that are used to define XML based languages. First among these is XML Schema - which has been adopted as a standard by the W3C. XML Schema addresses many of the limitations in DTDs, including:

Advanced datatyping, including datatyping for attributes.

Ability to include user defined meta-data in addition to standardized annotation and documentation.

XML Schema is written in XML, permitting manipulation by XML tools, such as XSLT, Xpath, etc.

FIXML 4.4 Schema Version Enhancements

The Schema version introduces the following enhancements

- Incorporated further transport optimizations
  - Adoption of attributes
  - Contextual Abbreviations – further reducing field names
- Addressed component blocks built around limitations of FIX tag=value by using consistent field names across component blocks
  - InstrumentLeg, NestedParties, Nested2Parties,UnderlyingInstrument
- Develop XML Schema Design Approach
  - Leverage work already done by ISO/XML and FpML
  - Design to support extensibility (customization) capabilities provided by FIX tag=value syntax

FIXML 4.4 Schema Version Design Objectives

Design objectives for FIXML messages (instance documents)

These design objectives refer to the FIXML instance documents. Instance documents are the actual FIXML messages.

- FIXML implementation shall adhere to XML technology standards as specified by the W3C.
  - FIXML implementation shall be suitable implementation for use in high volume transaction scenarios. Target applications:
    - Order Routing
    - Trade Reporting and Post Trade Processing
    - Distribution of product (instrument) information
    - Market making for lower volume applications
  - FIXML implementation shall minimize bandwidth consumption (reduced message size). The goal is to have FIXML messages be less than 1.5 X the size of an equivalent FIX tag=value message.
  - FIXML implementation shall maintain human readability of FIXML message, while still adhering to performance goals.
  - FIXML implementation shall support integration of FpML product specifications within the FIXML message in an equivalent manner to FIX 4.4 tag=value. This integration should use commonly agreed upon, de facto standard XML design patterns.
  - FIXML implementation shall support a ready translation to and from FIX tag=value messages.
  - FIXML implementation shall provide a cross-reference to ISO 15022 repository for each message, element, and component.
  - FIXML implementation shall maintain the extensibility and customization available via the FIX tag=value message format, including:
    - Ability to add custom messages,

- Ability to add custom fields to messages, component blocks, and repeating groups.

- FIXML Implementation shall provide full transport level independence.
- FIXML Implementation shall support version identification.

Design Objectives for the Schema Document

- FIXML Schema shall be implemented using the current de facto industry best practices for XML Schema usage.
- FIXML Schema shall be implemented in such a way as to fully support the FIXML 4.4 "Schema Version" Instance Requirements defined above.
- FIXML Schema shall support version identification.
- FIXML Schema shall provide meta-data sufficient to identify the FIX field name, component type, tag number, ISO 15022 repository cross-reference.
- FIXML Schema shall be interoperable and compatible with the FpML schema.
- The FIXML Schema shall be based upon and be compatible with the current version of XML schema: ***http://www.w3.org/2001/XMLSchema***

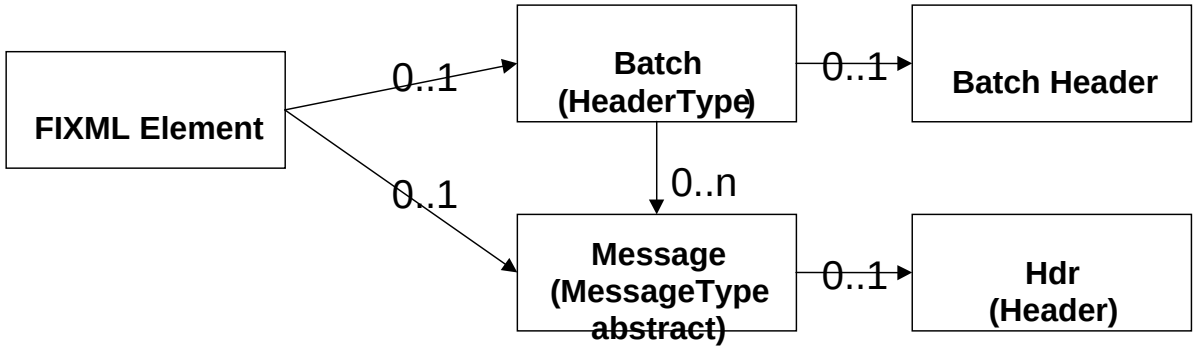
FIXML Design Rules

The following design guidelines were created to meet the design objectives for the FIXML Schema and the FIXML instance documents defined above.

1. Use meaningful abbreviations for element and attribute names wherever possible. Use standard abbreviations for common words (e.g., Price = Px, Currency = Ccy, etc.).
2. FIX Messages shall be implemented as XML Elements.
3. Individual, non-repeating fields shall be implemented as attributes of FIX Message elements.
4. FIX Component Blocks shall be implemented as an XML element.
5. Component blocks that were duplicated within FIX to circumvent tag=value requirements for uniqueness across fields and tag numbers, such as the Parties, NestedParties, NestedParties2 component blocks, shall use common naming in FIXML. The datatypes for each of the ComponentTypes will provide the mapping back to FIX tag=value format.
6. Non-repeating fields belonging to a FIX component block shall be implemented as attributes.
7. Repeating groups shall be implemented as XML elements.
8. Non-repeating fields belonging to a repeating group shall be implemented as attributes.
9. Identical repeating groups that occur across FIX messages will be identified as implicit components and reused across messages.
10. Field name prefixes that were used in FIX tag=value format for uniqueness shall be removed – thus creating a contextual abbreviation.
11. FIX datatypes will be mapped to the closest XML Schema datatype whenever possible, thus making FIXML more compatible with standard XML toolsets.

FIXML Schema Root Element

The FIXML Schema root element has been expanded to include the ability to include a batch of FIXML application messages. Batch capability was provided to deliver groups of messages, such as post trade confirms or position reports at the end of a trading session. Single message capability is still supported. Note that the headres are optional.



Single Message Usage

```
<FIXML>
    <Order>
       <Hdr/>
     </Order>
</FIXML>
```

Batch Message Usage

```
<FIXML>
   <Batch>
      <Hdr/>
      <Order>
        <Hdr/>
      </Order>
      <Order>
         <Hdr/>
      </Order>
    </Batch>
</FIXML>
```

An Example FIXML Single Message

The following is a New Order Single FIXML Schema message sent individually.

```
<FIXML v="4.4" r="20030618" s="20040109">
   <Order ClOrdID="123456" Side="2" TransactTm="2001-09-11T09:30:47-05:00"
OrdTyp="2" Px="93.25" Acct="26522154">
      <Instrmt Sym="IBM" ID="459200101" IDSrc="1"/>
      <OrdQty Qty="1000"/>
   </Order>
</FIXML>
```


An Example FIXML Batch Message

The following example shows a batch of position reports.

Note that the header is provided for the entire batch of messages.

```
<FIXML v="4.4" r="20030618" s="20031030">
   <Batch>
            <Hdr Snt="2001-12-17T09:30:47-05:00">
          <Sndr ID="OCC"/>
          <Tgt ID="Firm"/>
        </Hdr>
        <PosRpt  RptID="541386431"  Rslt="0"  BizDt="2003-09-10T00:00:00"  Acct="1"
AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
               <Pty ID="OCC" Role="21"/>
               <Pty ID="99999" Role="4"/>
            <Pty ID="C" Role="38">
             <PtySub SubID="ZZZ" SubIDTyp="2"/>
                  </Pty>
                  <Qty Typ="SOD" Long="35" Short="0"/>
                  <Qty Typ="FIN" Long="20" Short="10"/>
                  <Qty Typ="IAS" Long="10"/>
                  <Amt Typ="FMTM" Amt="0.00"/>
                 <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122"
Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
         </PosRpt>
        <PosRpt  RptID="541386536"  Rslt="0"  BizDt="2003-09-10T00:00:00"  Acct="1"
AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
               <Pty ID="OCC" Role="21"/>
               <Pty ID="99999" Role="4"/>
            <Pty ID="C" Role="38">
             <PtySub SubID="ZZZ" SubIDTyp="2"/>
                  </Pty>
                  <Qty Typ="SOD" Long="35" Short="0"/>
                  <Qty Typ="FIN" Long="20" Short="10"/>
                  <Qty Typ="IAS" Long="10"/>
                  <Amt Typ="FMTM" Amt="0.00"/>
                 <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122"
Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
         </PosRpt>
        <PosRpt  RptID="541386678"  Rslt="0"  BizDt="2003-09-10T00:00:00"  Acct="1"
AcctTyp="1" SetPx="0.00" SetPxTyp="1" PriSetPx="0.00" ReqTyp="0" Ccy="USD">
               <Pty ID="OCC" Role="21"/>
               <Pty ID="99999" Role="4"/>
            <Pty ID="C" Role="38">
             <PtySub SubID="ZZZ" SubIDTyp="2"/>
                  </Pty>
                  <Qty Typ="SOD" Long="35" Short="0"/>
                  <Qty Typ="FIN" Long="20" Short="10"/>
                  <Qty Typ="IAS" Long="10"/>
                  <Amt Typ="FMTM" Amt="0.00"/>
                 <Instrmt Sym="AOL" ID="KW" IDSrc="J" CFI="OCASPS" MMY="20031122"
Mat="2003-11-22T00:00:00" Strk="47.50" StrkCcy="USD" Mult="100"/>
         </PosRpt>
   </Batch>
</FIXML>
```

Version Identification

FIXML versions are identified explicitly in the schema file names and also with constant attribute values defined in the fixml-component-base schema file.

FIXML Schema File Versioning

FIXML Schema employed the file naming convention developed for FpML. The major and minor version numbers of the FIX version represented by the schema are appended to all FIXML schema file names. This approach was taken to explicitly force users to recognize when counterparties have changed their version of the schema.

FIXML Message Versioning

The FIXML root element <FIXML> contains three attributes that define the version of the message. The FIXML root element is defined in the **fixml-components-base** schema file.

Attribute	Description	Format	Example
v	FIX Version	N.N	4.4
r	FIX Version release date (used to designate errata releases between FIX versions)	YYYYMMDD	20030618
s	Schema Release (used to designate schema releases between errata releases)	YYYYMMDD	20031030

Example:

```
<FIXML v="5.0" r="20061024" s="20061026"> </FIXML>
```

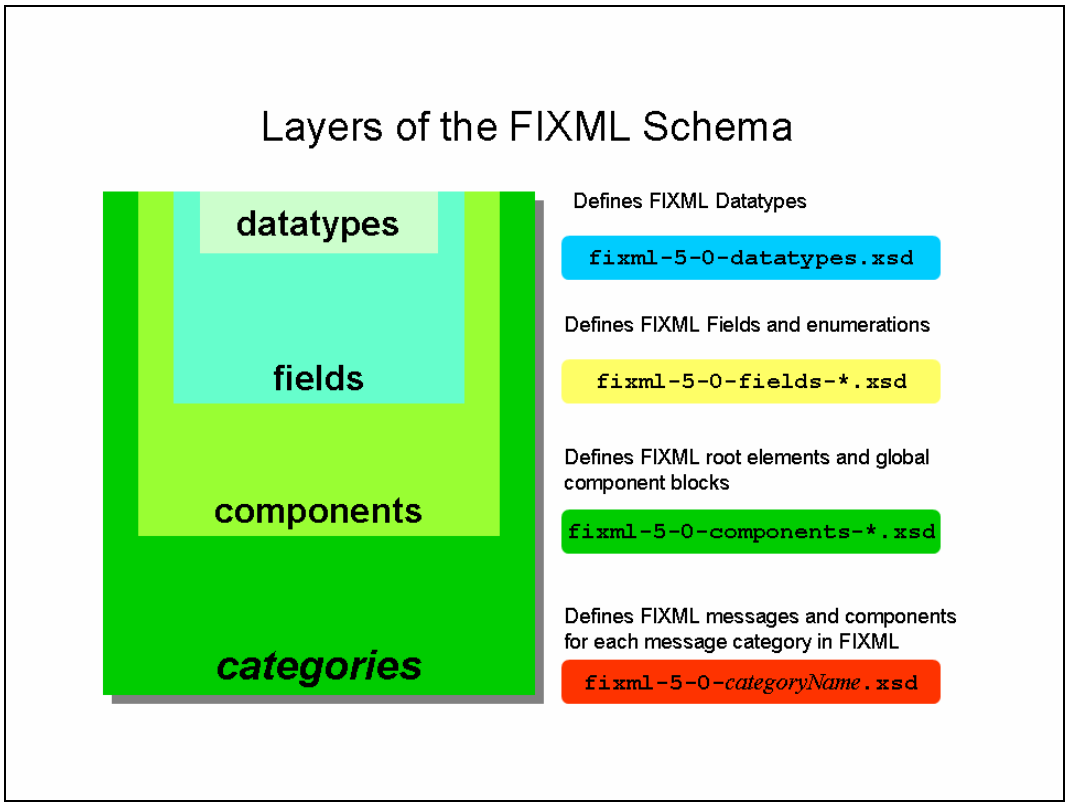
For FIX 5.0 changes have been made for versioning in order to be compatible with changes to support transport independence.

Version	FIXML Field	Abbreviation	FIX Tag	FIX Field Name	Discussion
FIX.4.4	Version	v	8	BeginString	Version of FIX
FIX.4.4	Release	r			Release date of FIX
FIX.4.4	SchemaRelease	s			Release date of the Schema
FIX.4.4	Extension Version	xv			Extension version
FIX.4.4	Extension Release	xr			Extension release date
New fields in the standard header					
FIX.5.0		v	1128	ApplVerID	Indicates application version using a service pack identifier. The ApplVerID applies to a specific message
FIX.5.0		r		deprecated	can be used to provide the version release date
FIX.5.0		xv	1129	CstmApplVerID	Used to support bilaterally agreed custom functionality
FIX.5.0		xr		deprecated	can be used to provide a release date for the extended version

FIXML Schema File Structure

Organization of files was driven largely by the requirement to support customization of the FIXML Schema per the requirements set forth by the FIXML Schema Working Group.

The basic organization of the schema has the datatypes used by the fields maintained in a separate file. FIX fields are defined in the shared file. Components and the FIXML root element are defined in the component files. FIXML messages are defined within separate category files.



Extensibility Design Pattern

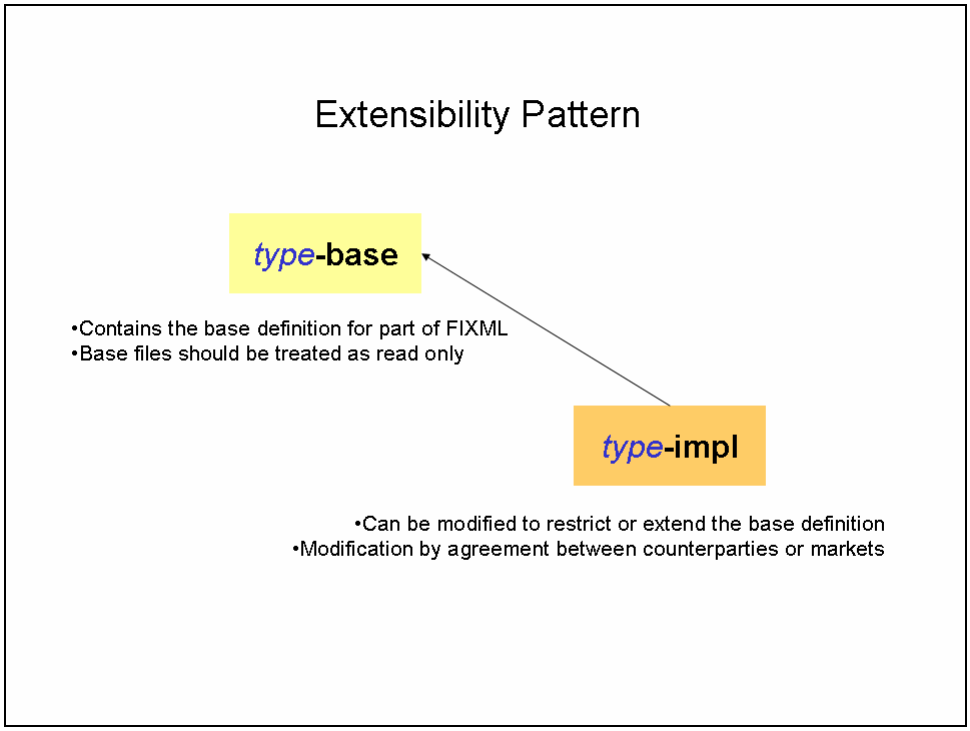
Much of the design work that went into the FIXML Schema was done to permit counterparties to further refine the FIXML language either by restriction or extension.

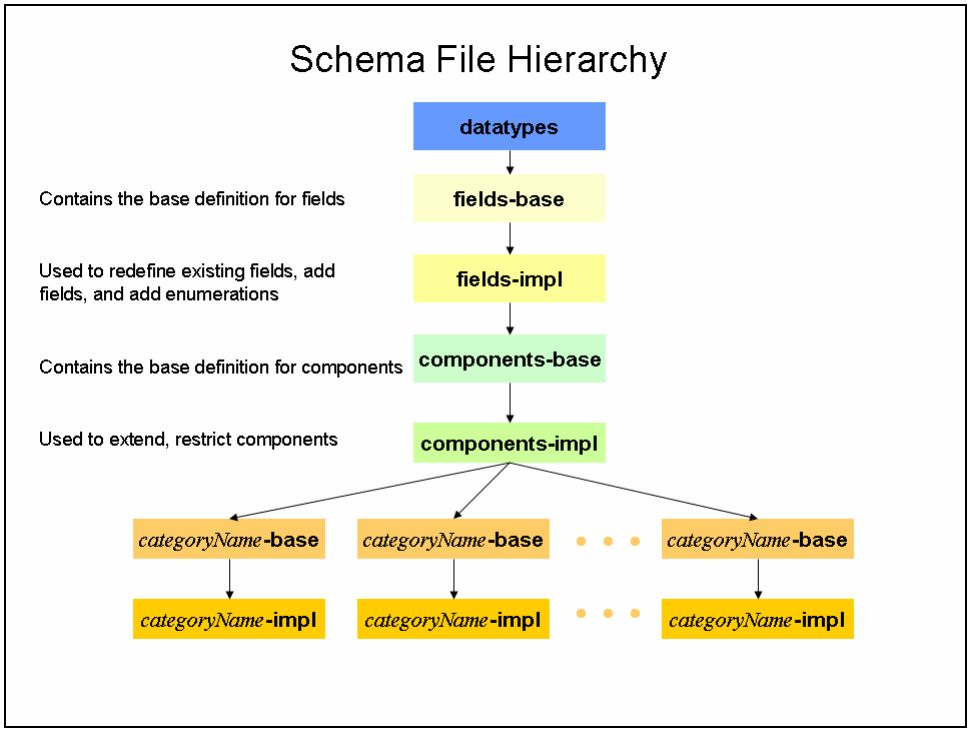
A possible scenario for restriction would be a market place that only supports a subset of the enumerations available for OrdType (tag=39). The exchange can override the OrdType_t FIXML datatype in the fixml-shared-impl-M-N.xsd file to restrict the set of possible values to only those supported by the market place.

An example ofextension would be counterparties that require an additional custom field to be added to a new message.

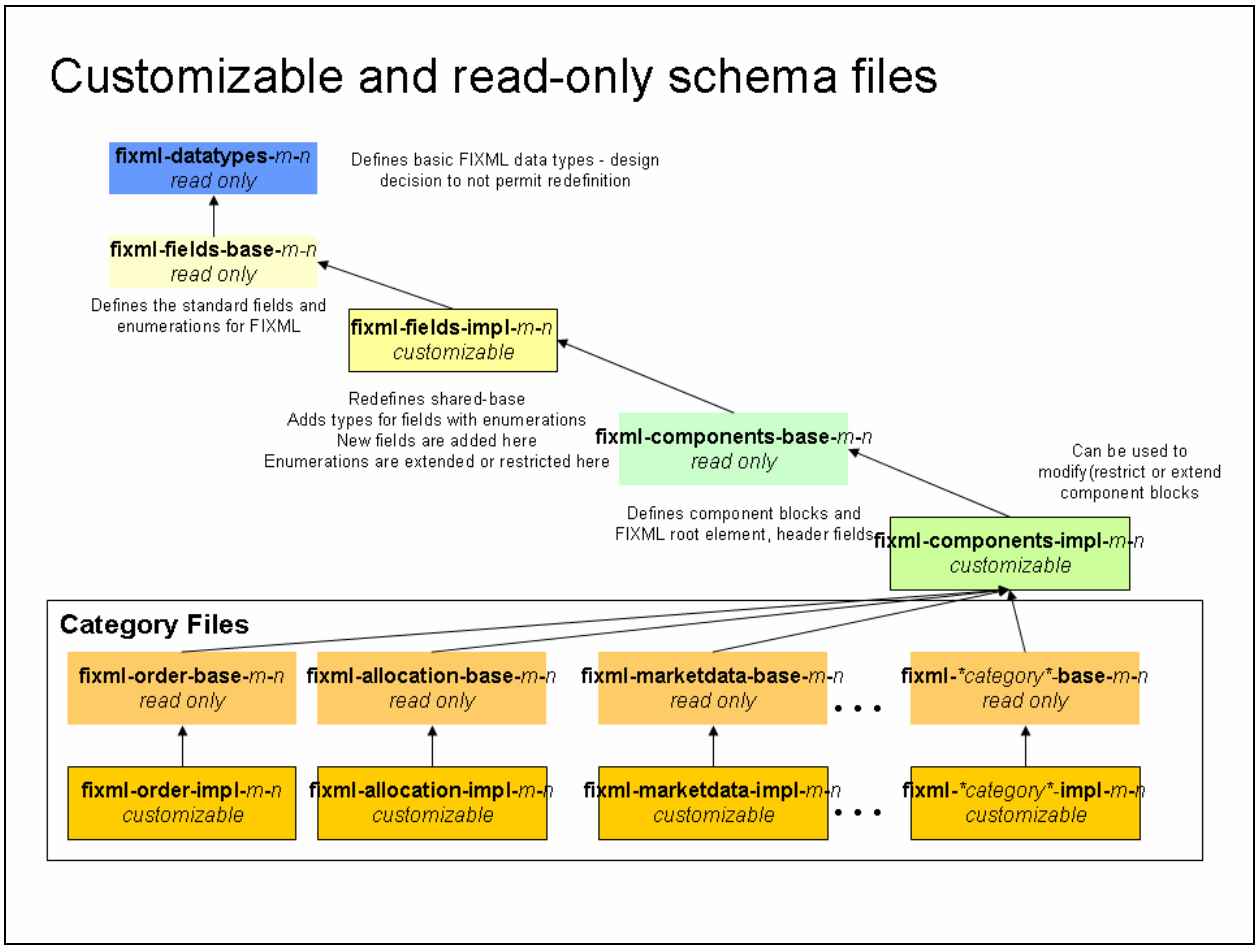
In order to provide a uniform method in defining customizations that could be readily absorbed by counterparties an extensibility design pattern was developed that defines how the FIXML definition was partitioned and organized within separate schema files.

Each level of schema file (with the exception of datatypes) provides a base definition file that defines the standard (default) FIXML language. Redefining this base file an implementation file (“impl”) is provided that by default simply references the base definition.





Subsequent levels of the schema reference the impl from the previous level – thus providing a customization entry point at the field level, component level, and message level.



FIXML Schema file naming conventions

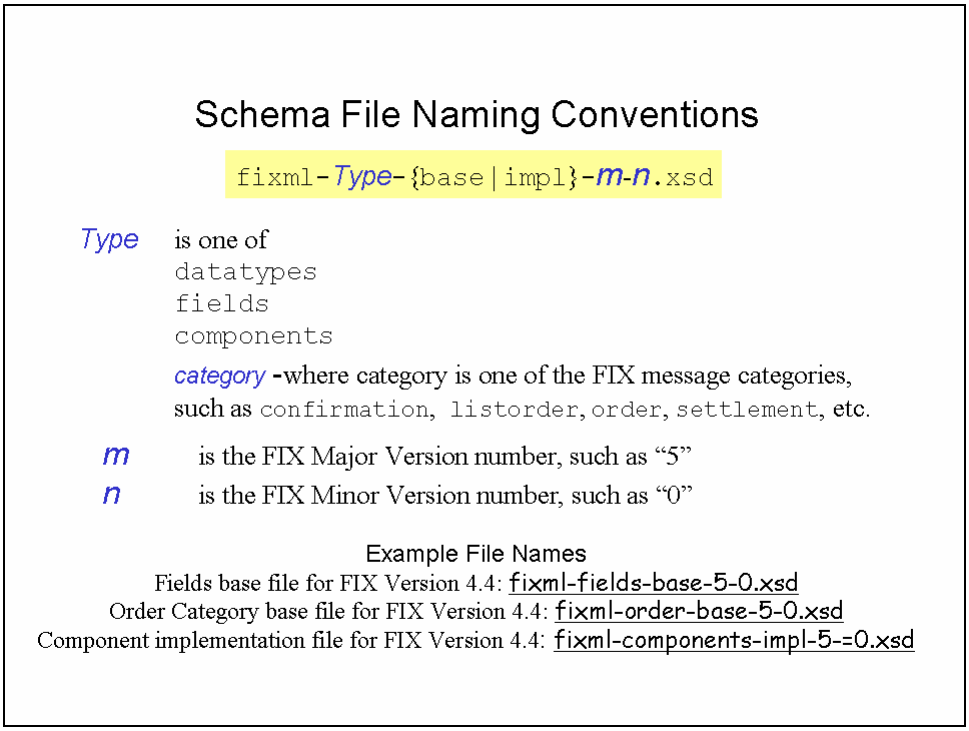
FIXML file naming conventions are shown in the followin illustration.

All filenames begin with lowercase “fixml-“

“-“ is used to separate portions of the filename

The type of the schema file is identified in the second component of the file name. The datatypes file contains the basic datatypes used within FIXML. The shared files contain the definitions for FIX fields. The components file contains definitions for FIXML components (as defined in Volume 1 of the specification, additional components identified while defining the FIXML schema, and the outer elements for FIX.

Files are either a **base** file or an implementation (**impl**). Base files define the standard FIXML language. Impl files are used to extend or restrict the base FIXML language.



Refer to the FIXML Schema File Summary section for a complete list of schema files used in FIXML as of FIX release 4.4.

Datatypes schema file

A decision was made to use native XML Schema datatypes wherever possible. Many of the XML Schema standards are based upon ISO standard datatypes. This means that the FIX representation of UTCTimestamp is different from the FIXML representation. The FIXML Schema working group felt it more important to be compatible with XML and as a result XML toolsets. The requirement for conversion between FIX tag=value datatypes and XML is left to implementors.

The **fixml-datatypes** schema file contains definitions for the FIXML datatypes.

FIX 5.0 introduces **pattern datatypes** that are used to appropriately support customization of enumerations and also to support types that require both enumerations and specific patterns, such as the SettlementType field. The <xs:union> element is used to combine an enumerated type with a pattern type in the fixml-fields-impl-M-N.xsd file..

The following patterns have been created to support validation of user defined enumeration values and extended patterns.

Tenor	Pattern	<xs:simpleType name="Tenor"><xs:restriction base="xs:string"> <xs:pattern value="[DMWY](\d)+"/> </xs:restriction> </xs:simpleType>	Currently used to support the SettlementType which can be either an enumeration or a tenor pattern, such as M6 (six month).
Reserved100Plus	Pattern	<xs:simpleType name="Reserved100Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="100"/> </xs:restriction> </xs:simpleType>	Used for enumerated fields that permit user defined values of 100 and greater.
Reserved1000Plus	Pattern	<xs:simpleType name="Reserved1000Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="1000"/> </xs:restriction> </xs:simpleType>	Used for enumerated fields that permit user defined values of 1000 and greater.
Reserved4000Plus	Pattern	<xs:simpleType name="Reserved4000Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="4000"/> </xs:restriction> </xs:simpleType>	Used for enumerated fields that permit user defined values of 4000 and great.

Example union types from fixml-fields-impl-M-N.xsd:

<xs:simpleType name="SettlType_t"> <xs:union memberTypes="SettlType_enum_t Tenor"/> </xs:simpleType>	The Settlement type is a union of the settlement type enumerations and the Tenor type described above
<xs:simpleType name="OrdRejReason_t"> <xs:union memberTypes="OrdRejReason_enum_t Reserved100Plus"/> </xs:simpleType>	The OrderRejectReason field is a union of the OrderReject Reason enumerations and can also be extended with user defined values of 100 or greater.

Fields schema files

- Fields schema file (fixml-fields-*-M-N.xsd)
- Fields base file (fixml-fields-base-M-N.xsd)

The **fixml-fields-base** file contains simple type definitions for all FIX application level fields and session level fields that are used as part of the FIXML header. All fields are defined as simple types. The simple type name is derived from the full FIX field name appended with a "_t". All fields with enumerations are defined as simple types. The enumeration simple type name is derived from the full FIX field name appended with a "enum_t".

Field definition examples

An example of a field definition for the AvgPx (tag=6) field:

```
<xs:simpleType name="AvgPx_t">
   <xs:annotation>
      <xs:documentation xml:lang="en">Calculated average price of all fills
on this order    For Fixed Income trades AvgPx is always expressed as
percent of par  regardless of the PriceType  423  of LastPx  3    I e
AvgPx will contain an average of percent of par values  see LastParPx  669
for issues traded in Yield  Spread or Discount
      </xs:documentation>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <xs:Xref Protocol="FIX" name="AvgPx" tag="6" datatype="Price"
ComponentType="Field"/>
         <xs:Xref Protocol="ISO_15022_XML"/>
      </xs:appinfo>
   </xs:annotation>
   <xs:restriction base="Price"/>
</xs:simpleType>
```

An example of an enumerated field:

```
<xs:simpleType name="CommType_enum_t">
   <xs:annotation>
      <xs:documentation xml:lang="en">Commission type Valid values:  = per
unit  implying shares  par  currency  etc   2 = percentage 3 = absolute
total monetary amount  4 =   for CIV buy orders  percentage waived   cash
discount 5 =   for CIV buy orders  percentage waived   enhanced units 6 =
points per bond or or contract   Supply ContractMultiplier  23  in the
Instrument  component block if the object security is denominated in a size
other than the industry default   000 par for bonds
      </xs:documentation>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <xs:Xref Protocol="FIX" name="CommType" tag="13" datatype="char"
ComponentType="Field"/>
         <xs:Xref Protocol="ISO_15022_XML"/>
      </xs:appinfo>
      <xs:appinfo xmlns:x="http://www.fixprotocol.org/fixml/metadata.xsd">
         <x:EnumDoc value="1" desc="PerShare"/>
         <x:EnumDoc value="2" desc="Percent"/>
         <x:EnumDoc value="3" desc="Absolute"/>
         <x:EnumDoc value="4" desc="PctWaivedCshDisc"/>
         <x:EnumDoc value="5" desc="PctWaivedEnUnits"/>
         <x:EnumDoc value="6" desc="PerBond"/>
      </xs:appinfo>
   </xs:annotation>
   <xs:restriction base="xs:string">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
      <xs:enumeration value="3"/>
      <xs:enumeration value="4"/>
      <xs:enumeration value="5"/>
      <xs:enumeration value="6"/>
   </xs:restriction>
</xs:simpleType>
```


Fields implementation file (fixml-fields-impl-M-N.xsd)

One of the more convoluted constructs used was the need to place the field level definitions for enumerated types in the **fixml-fields-impl** file. As shown above, the **fixml-fields-base** file defines each enumerated field as a simple type named ***fieldname*_enum_t**. This enumerated type is then used to define a corresponding field type in the **fixml-fields-impl** schema file named ***fieldname*_t**. It is this ***fieldname*_t** type that is referenced in subsequent schema files (fixml-components and the message category schema files). This construct was required to provide a mechanism to extend enumerations. The ***fieldname*_t** can be modified in the **fixml-fields-impl** file to include additional enumerations. The ***fieldname*_t** can be restricted by redefining the ***fieldname*_enum_t** simple type within the fixed-shared-impl file.

Components (fixml-components-*-M-N.xsd)

Component files are used to define the reusable components that are used across FIX messages. The FIXML root element and headers are defined in the components file, as well.

Components base file (fixml-components-base-M-N.xsd

The **fixml-components-base file** contains the definitions for all FIX component blocks defined in volume 1 of the FIX specification. The FIXML root element, FIXML headers, the batch element, and the abstract message type are also defined within this file.

Components (and messages) are defined using element groups and attribute groups. The advantage of these groups is that you can redefine the groups (using either restriction or extension) to change the overall structure of the component (or message).

These groups are defined for each component and message.

*componentOrMessageName*Elements	Contains a list of elements contained in the component.
*componentOrMessageName*Attributes	Contains a list of Attributes contained in the component.

The Parties Component block is shown below. Notice the overall definition pattern. This pattern is followed for all component blocks and message definitions.

```
        <xs:group name="PartiesElementsRequired">
               <xs:sequence/>
        </xs:group>
        <xs:group name="PartiesElementsOptional">
               <xs:sequence>
               <xs:element name="PtySub" type="PtysSubGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
               </xs:sequence>
        </xs:group>
        <xs:group name="PartiesElementsCustom">
               <xs:sequence/>
        </xs:group>
        <xs:attributeGroup name="PartiesAttributesRequired">

        </xs:attributeGroup>
        <xs:attributeGroup name="PartiesAttributesOptional">
               <xs:attribute name="ID" type="PartyID_t" use="optional"/>
               <xs:attribute name="IDSrc" type="PartyIDSource_t" use="optional"/>
               <xs:attribute name="Role" type="PartyRole_t" use="optional"/>
        </xs:attributeGroup>
        <xs:attributeGroup name="PartiesAttributesCustom"/>

        <xs:complexType name="Parties_Block_t" final="#all">
        <xs:annotation>
               <xs:documentation xml:lang="en">**Desc**
               </xs:documentation>
               <xs:appinfo>
                      <fm:Xref Protocol="FIX" name="Parties"
ComponentType="BlockRepeating"/>
                      <xs:Xref Protocol="ISO_15022_XML"/>
               </xs:appinfo>
        </xs:annotation>
               <xs:sequence>
                      <xs:group ref="PartiesElementsRequired"/>
                      <xs:group ref="PartiesElementsOptional"/>
                      <xs:group ref="PartiesElementsCustom"/>
               </xs:sequence>
               <xs:attributeGroup ref="PartiesAttributesRequired"/>
               <xs:attributeGroup ref="PartiesAttributesOptional"/>
               <xs:attributeGroup ref="PartiesAttributesCustom"/>
        </xs:complexType>
```

Components implementation file (fixml-components-impl-M-N.xsd)

The default version **fixml-components-impl** file simply redefines the components-base file. This is the file where modifications (restrictions or extensions) would be made to component blocks used in the FIX protocol.

Categories (fixml-categoryName-base-M-N.xsd)

Each message category defined within the FIX specification has its own schema file. This provides a granular level of usage for applications only requiring access to one message categoryThe message category schema files contain the component and message definitions that belong to a specific message category defined within the FIX Protocol. Examples of message categories include: Indications, Market Data, Positions, Allocation. . A complete list of the category files for FIXML is provided in the FIXML Schema File Summary section.

Category messages and components are defined following the same pattern defined above for components. The following defines the New Order Single message from the fixml-categoryOrder-5-0.xsd:

```
   <xs:group name="NewOrderSingleElementsRequired">
         <xs:sequence>
         <xs:element name="Instrmt" type="Instrument_Block_t" minOccurs="1"
maxOccurs="1"/>
         <xs:element name="OrdQty" type="OrderQtyData_Block_t" minOccurs="1"
maxOccurs="1"/>
         </xs:sequence>
   </xs:group>
   <xs:group name="NewOrderSingleElementsOptional">
         <xs:sequence>
         <xs:element name="Pty" type="Parties_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="Stip" type="Stipulations_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="FinDetls" type="FinancingDetails_Block_t" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="SprdBnchmkCurve"
type="SpreadOrBenchmarkCurveData_Block_t" minOccurs="0" maxOccurs="1"/>
         <xs:element name="Yield" type="YieldData_Block_t" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="Comm" type="CommissionData_Block_t" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="PegInstr" type="PegInstructions_Block_t" minOccurs="0"
maxOccurs="1"/>
         <xs:element name="DiscInstr" type="DiscretionInstructions_Block_t"
minOccurs="0" maxOccurs="1"/>
         <xs:element name="PreAll" type="PreAllocGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="TrdSes" type="TrdgSesGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
         <xs:element name="Undl" type="UndInstrmtGrp_Block_t" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
   </xs:group>
   <xs:group name="NewOrderSingleElementsCustom">
         <xs:sequence/>
   </xs:group>
   <xs:attributeGroup name="NewOrderSingleAttributesRequired">
         <xs:attribute name="ClOrdID" type="ClOrdID_t" use="required"/>
         <xs:attribute name="Side" type="Side_t" use="required"/>
         <xs:attribute name="TransactTm" type="TransactTime_t" use="required"/>
         <xs:attribute name="OrdTyp" type="OrdType_t" use="required"/>

   </xs:attributeGroup>
   <xs:attributeGroup name="NewOrderSingleAttributesOptional">
         <xs:attribute name="ScndClOrdID" type="SecondaryClOrdID_t" use="optional"/>
         <xs:attribute name="ClOrdLinkID" type="ClOrdLinkID_t" use="optional"/>
         <xs:attribute name="TrdOrigntnDt" type="TradeOriginationDate_t"
use="optional"/>
         <xs:attribute name="TrdDt" type="TradeDate_t" use="optional"/>
         <xs:attribute name="Acct" type="Account_t" use="optional"/>
         <xs:attribute name="AcctIDSrc" type="AcctIDSource_t" use="optional"/>
         <xs:attribute name="AcctTyp" type="AccountType_t" use="optional"/>
         <xs:attribute name="DayBkngInst" type="DayBookingInst_t" use="optional"/>
         <xs:attribute name="BkngUnit" type="BookingUnit_t" use="optional"/>
```



```
        <xs:attribute name="PreallocMethod" type="PreallocMethod_t"
use="optional"/>
        <xs:attribute name="AllocID" type="AllocID_t" use="optional"/>
        <xs:attribute name="SettlTyp" type="SettlType_t" use="optional"/>
        <xs:attribute name="SettlDt" type="SettlDate_t" use="optional"/>
        <xs:attribute name="CshMgn" type="CashMargin_t" use="optional"/>
        <xs:attribute name="ClrngFeeInd" type="ClearingFeeIndicator_t"
use="optional"/>
        <xs:attribute name="HandlInst" type="HandlInst_t" use="optional"/>
        <xs:attribute name="ExecInst" type="ExecInst_t" use="optional"/>
        <xs:attribute name="MinQty" type="MinQty_t" use="optional"/>
        <xs:attribute name="MaxFloor" type="MaxFloor_t" use="optional"/>
        <xs:attribute name="ExDest" type="ExDestination_t" use="optional"/>
        <xs:attribute name="ProcCode" type="ProcessCode_t" use="optional"/>
        <xs:attribute name="PrevClsPx" type="PrevClosePx_t" use="optional"/>
        <xs:attribute name="LocReqd" type="LocateReqd_t" use="optional"/>
        <xs:attribute name="QtyTyp" type="QtyType_t" use="optional"/>
        <xs:attribute name="PxTyp" type="PriceType_t" use="optional"/>
        <xs:attribute name="Px" type="Price_t" use="optional"/>
        <xs:attribute name="StopPx" type="StopPx_t" use="optional"/>
        <xs:attribute name="Ccy" type="Currency_t" use="optional"/>
        <xs:attribute name="ComplianceID" type="ComplianceID_t" use="optional"/>
        <xs:attribute name="SolFlag" type="SolicitedFlag_t" use="optional"/>
        <xs:attribute name="IOIID" type="IOIID_t" use="optional"/>
        <xs:attribute name="QID" type="QuoteID_t" use="optional"/>
        <xs:attribute name="TmInForce" type="TimeInForce_t" use="optional"/>
        <xs:attribute name="EfctvTm" type="EffectiveTime_t" use="optional"/>
        <xs:attribute name="ExpireDt" type="ExpireDate_t" use="optional"/>
        <xs:attribute name="ExpireTm" type="ExpireTime_t" use="optional"/>
        <xs:attribute name="GTBkngInst" type="GTBookingInst_t" use="optional"/>
        <xs:attribute name="Cpcty" type="OrderCapacity_t" use="optional"/>
        <xs:attribute name="Rstctions" type="OrderRestrictions_t" use="optional"/>
        <xs:attribute name="CustOrdCpcty" type="CustOrderCapacity_t"
use="optional"/>
        <xs:attribute name="ForexReq" type="ForexReq_t" use="optional"/>
        <xs:attribute name="SettlCcy" type="SettlCurrency_t" use="optional"/>
        <xs:attribute name="BkngTyp" type="BookingType_t" use="optional"/>
        <xs:attribute name="Txt" type="Text_t" use="optional"/>
        <xs:attribute name="EncTxtLen" type="EncodedTextLen_t" use="optional"/>
        <xs:attribute name="EncTxt" type="EncodedText_t" use="optional"/>
        <xs:attribute name="SettlDt2" type="SettlDate2_t" use="optional"/>
        <xs:attribute name="Qty2" type="OrderQty2_t" use="optional"/>
        <xs:attribute name="Px2" type="Price2_t" use="optional"/>
        <xs:attribute name="PosEfct" type="PositionEffect_t" use="optional"/>
        <xs:attribute name="CoveredOrUncovered" type="CoveredOrUncovered_t"
use="optional"/>
        <xs:attribute name="MaxShow" type="MaxShow_t" use="optional"/>
        <xs:attribute name="TgtStrategy" type="TargetStrategy_t" use="optional"/>
        <xs:attribute name="TgtStrategyParameters"
type="TargetStrategyParameters_t" use="optional"/>
        <xs:attribute name="ParticipationRt" type="ParticipationRate_t"
use="optional"/>
        <xs:attribute name="CxllationRights" type="CancellationRights_t"
use="optional"/>
        <xs:attribute name="MnyLaunderingStat" type="MoneyLaunderingStatus_t"
use="optional"/>
        <xs:attribute name="RegistID" type="RegistID_t" use="optional"/>
        <xs:attribute name="Designation" type="Designation_t" use="optional"/>

</xs:attributeGroup>
  <xs:attributeGroup name="NewOrderSingleAttributesCustom"/>

  <xs:complexType name="NewOrderSingle_message_t" final="#all">
        <xs:complexContent>
          <xs:extension base="Abstract_message_t">
             <xs:sequence>
                <xs:group ref="NewOrderSingleElementsRequired"/>
```

```
                <xs:group ref="NewOrderSingleElementsOptional"/>
                <xs:group ref="NewOrderSingleElementsCustom"/>
            </xs:sequence>
            <xs:attributeGroup ref="NewOrderSingleAttributesRequired"/>
            <xs:attributeGroup ref="NewOrderSingleAttributesOptional"/>
            <xs:attributeGroup ref="NewOrderSingleAttributesCustom"/>

         </xs:extension>
       </xs:complexContent>
  </xs:complexType>

<xs:element name="NewOrdSingle" type="NewOrderSingle_message_t"
substitutionGroup="Message" final="#all"/>
```

Categories (fixml-categoryName-impl-M-N.xsd)

Each message category defined within the FIX specification has its own schema file. This provides a granular level of usage for applications only requiring access to one message category. A complete list of the category files for FIXML is provided below in the FIXML File Summary table.

Trading Life Cycle files

Convenience files are provided with the FIXML schema version that includes the message categories for each of the trade life cycles (pre-trade, trade, post-trade) used by FIX. These files are provided to make it easier for applications that require access to multiple message categories within one of the trading life cycles.

Pretrade file (fixml-pretrade-M-N.xsd)

Includes the pre-trade message category implementation files.

Trade file (fixml-trade-M-N.xsd)

Includes the trade message category implementation files.

Post trade file (fixml-trade-M-N.xsd)

Includes the post trade message category implementation files.

Main (fixml-main-M-N.xsd)

A main schema file is included that pulls in the pretrade, trade, and post trade schema files. This is provided for applications that require access to the full suite of FIX messages.

Customization

The FIXML Schema files have been organized to permit extensibility. Implementation versions of each schema file (with the exception of the datatypes file) are provided to permit users to redefine the base FIXML Schema version, as defined in the base files. This section provides guidelines for customizing the FIXML syntax. Even though a considerable amount of work has gone into making FIXML extensible, users are strongly encouraged to minimize modifications, in order to promote more consistent usage of the FIXML syntax within the industry. Obviously, the less customization, the easier it is to connect to counterparties. If customization is required, you are encouraged to communicate your requirements that are not being met by FIX to the FPL Global Technical Committee. There you may find out that there is a technique to meet your business requirement. Or, you may

find that the Technical Committee has already addressed the issue for a planned future release. At a minimum you will receive coaching and assistance in how to extend FIXML in such a way as to make the new feature a part of a future version of FIX.

Defining a custom field

New fields are defined as an XML SimpleType in the fixml-shared-impl-N-N.xsd file. You are recommended to add the file to the end of the schema document. You also are strongly encouraged to include XML comments to define the reason for the field.

The field should then be added to the component or message where it will be used, once the field is defined in the fixml-shared-impl schema file.

If the field will be added to a component contained in fixml-components-base-N-N.xsd, you must now redefine that component in the fixml-components-impl-N-N.xsd file.

Adding a field to a component or message contained in one of the message categories is done in the same way you modify the components schema file. You need to redefine the portion of the message in the implementation version of the file.

You are encouraged to follow the same procedure for procuring new custom field names as is done for the FIX tag=value version of FIX. The FIX website provides a web page of custom fields and a form to submit requests for additional custom fields.

Restricting enumeration values for a FIX field

Restricting enumeration values is done by modifying the type definition in the fixml-shared-impl schema file.

Extending enumeration values for a FIX field

Extending enumeration values is done by creating a union of the original enumeration type definition with new enumeration values.

Making an optional field required

Making an optional field required is done by redefining the optional attribute group, modifying the usage of the field from “optional” to “required”. This redefinition is done within the implementation file for either the components or a particular message category.

Making a required field optional

It is not possible to make a required field optional without modifying the original required element or attribute group. Making required fields optional does go against the standard base definition of FIX and should be avoided.

Adding a custom message

Custom messages are added by creating a message structure within the category to which the custom message belongs. Required and optional element and attribute groups should be created for the custom message.

FIXML Schema Version Datatypes

Type	Base Type	FIXML Implementation	Example
int		Use builtin type: xs:integer	
Length	int	<xs:simpleType name="Length"><xs:restriction base="xs:nonNegativeInteger"> </xs:restriction> </xs:simpleType>	
TagNum	int	NOT REQUIRED IN FIXML	
SeqNum	int	<xs:simpleType name="SeqNum"><xs:restriction base="xs:positiveInteger"> </xs:restriction> </xs:simpleType>	
NumInGroup	int	NOT REQUIRED IN FIXML	
float		Use builtin type: xs:decimal	
Qty	float	<xs:simpleType name="Qty"><xs:restriction base="xs:decimal"> </xs:restriction> </xs:simpleType>	
Price	float	<xs:simpleType name="Price"><xs:restriction base="xs:decimal"> </xs:restriction> </xs:simpleType>	Strk="47.50"
PriceOffset	float	<xs:simpleType name="PriceOffset"><xs:restriction base="xs:decimal"> </xs:restriction> </xs:simpleType>	
Amt	float	<xs:simpleType name="Amt"><xs:restriction base="xs:decimal"> </xs:restriction> </xs:simpleType>	Amt="6847.00"
Percentage	float	<xs:simpleType name="Percentage"><xs:restriction base="xs:decimal"> </xs:restriction> </xs:simpleType>	
char		<xs:simpleType name="char"><xs:restriction base="xs:string"> <xs:pattern value=".{1}"/> </xs:restriction> </xs:simpleType>	
Boolean	char	Use builtin type: xs:boolean	
String		Use builtin type: xs:string	
MultipleCharValue	String	<xs:simpleType name="MultipleCharValue"><xs:restriction base="xs:string"> <xs:pattern value="[A-Za-z0-9](\s[A-Za-z0-9])*"/> </xs:restriction> </xs:simpleType>	

Type	Base Type	FIXML Implementation	Example
MultipleStringValue	String	<xs:simpleType name="MultipleStringValue"><xs:restriction base="xs:string"> <xs:pattern value="([A-Za-z0-9])+(\s([A-Za-z0-9])+)*"/> </xs:restriction> </xs:simpleType>	
Country	String	<xs:simpleType name="Country"><xs:restriction base="xs:string"> <xs:pattern value=".{2}"/> </xs:restriction> </xs:simpleType>	
Currency	String	<xs:simpleType name="Currency"><xs:restriction base="xs:string"> <xs:pattern value=".{3}"/> </xs:restriction> </xs:simpleType>	StrkCcy="USD"
Exchange	String	<xs:simpleType name="Exchange"><xs:restriction base="xs:string"> <xs:pattern value=".*"/> </xs:restriction> </xs:simpleType>	
MonthYear	String	<xs:simpleType name="MonthYear"><xs:restriction base="xs:string"> <xs:pattern value="\d{4}(0|1)\d([0-3wW]\d)?"/> </xs:restriction> </xs:simpleType>	MonthYear="200303", MonthYear="20030320", MonthYear="200303w2"
UTCTimestamp	String	<xs:simpleType name="UTCTimestamp"><xs:restriction base="xs:dateTime"> </xs:restriction> </xs:simpleType>	TransactTm="2001-12-17T09:30:47-05:00"
UTCTimeOnly	String	<xs:simpleType name="UTCTimeOnly"><xs:restriction base="xs:time"> </xs:restriction> </xs:simpleType>	MDEntryTime="13:20:00.000-05:00"
UTCDateOnly	String	<xs:simpleType name="UTCDateOnly"><xs:restriction base="xs:date"> </xs:restriction> </xs:simpleType>	MDEntryDate="2003-09-10"
LocalMktDate	String	<xs:simpleType name="LocalMktDate"><xs:restriction base="xs:date"> </xs:restriction> </xs:simpleType>	BizDate="2003-09-10"
TZTimeOnly	String	<xs:simpleType name="TZTimeOnly"><xs:restriction base="xs:time"> </xs:restriction> </xs:simpleType>	
TZTimestamp	String	<xs:simpleType name="TZTimestamp"><xs:restriction base="xs:dateTime"> </xs:restriction> </xs:simpleType>	

Type	Base Type	FIXML Implementation	Example
data	String	<xs:simpleType name="data"><xs:restriction base="xs:string"> </xs:restriction> </xs:simpleType>	
Pattern		NOT REQUIRED IN FIXML	
Tenor	Pattern	<xs:simpleType name="Tenor"><xs:restriction base="xs:string"> <xs:pattern value="[DMWY](\d)+"/> </xs:restriction> </xs:simpleType>	
Reserved100Plus	Pattern	<xs:simpleType name="Reserved100Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="100"/> </xs:restriction> </xs:simpleType>	
Reserved1000Plus	Pattern	<xs:simpleType name="Reserved1000Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="1000"/> </xs:restriction> </xs:simpleType>	
Reserved4000Plus	Pattern	<xs:simpleType name="Reserved4000Plus"><xs:restriction base="xs:integer"> <xs:minInclusive value="4000"/> </xs:restriction> </xs:simpleType>	

FIXML Schema File Summary

File Name	Description
Fixml-datatypes-5-0.xsd	Defines the base data types that are to be used in other fixml schema files. These fixml base data types are based on simple types built into XML Schema.
Fixml-fields-base-5-0.xsd	Includes fixml-datatypes-5-0.xsd. Defines simple/complex types for each field value based on fixml-datatypes-5-0.xsd
Fixml-fields-impl-5-0.xsd	Includes fixml-datatypes-base-5-0.xsd. Derived types can be added here in the redefine.
Fixml-components-base-5-0.xsd	Includes fixml-fields-impl-5-0.xsd. Defines groups and attributeGroups that make up the complexTypes used by message schema to define that message.
Fixml-components-impl-5-0.xsd	Includes fixml-components-base-5-0.xsd. Derived types can be added here in the redefine.

Fixml-allocation-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines allocation messages: AllocationInstruction AllocationInstructionAck AllocationReport AllocationReportAck AllocationInstructionAlert
Fixml-allocation-impl-5-0.xsd	Includes fixml-allocation-base-5-0.xsd Used to customize allocation message category
Fixml-collateral-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines collateral messages: CollateralRequest CollateralAssignment CollateralResponse CollateralReport CollateralInquiry CollateralInquiryAck
Fixml-collateral-impl-5-0.xsd	Includes fixml-collateral-base-5-0.xsd Used to customize collateral message category
Fixml-confirmation-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines confirmation messages: Confirmation Confirmation_Ack ConfirmationRequest
Fixml-confirmation-impl-5-0.xsd	Includes fixml-confirmation-base-5-0.xsd Used to customize confirmation message category
Fixml-crossorders-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines cross orders messages: NewOrderCross CrossOrderCancelReplaceRequest CrossOrderCancelRequest
Fixml-crossorders-impl-5-0.xsd	Includes fixml-crossorders-base-5-0.xsd Used to customize crossorders message category
Fixml-indications-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines indications messages: IOI Advertisement
Fixml-indications-impl-5-0.xsd	Redefines fixml-indications-base-5-0.xsd Used to customize indication message category

Fixml-listorders-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines list orders messages: NewOrderList ListCancelRequest ListExecute ListStatusRequest ListStatus BidRequest BidResponse ListStrikePrice
Fixml-listorders-impl-5-0.xsd	Includes fixml-listorders-base-5-0.xsd Used to customize list orders message category
Fixml-multilegorders-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines multiple go orders messages: NewOrderMultileg MultilegOrderCancelReplace NewOrderMultileg MultilegOrderCancelReplace NewOrderMultileg MultilegOrderCancelReplace
Fixml-multilegorders-impl-5-0.xsd	Includes fixml-multilegorders-base-5-0.xsd Used to customize multileg orders message category
Fixml-marketdata-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines trade market data messages: MarketDataRequest MarketDataSnapshotFullRequest MarketDataIncrementalRequest MarketDataRequestReject
Fixml-marketdata-impl-5-0.xsd	Redefines fixml-marketdata-base-5-0.xsd Used to customize marketdata message category

Fixml-order-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines order messages: ExecutionReport OrderCancelReject NewOrderSingle OrderCancelRequest OrderCancelReplaceRequest OrderStatusRequest DontKnowTradeDK OrderMassCancelRequest OrderMassCancelReport OrderMassStatusRequest ExecutionAcknowledgement
Fixml-orders-impl-5-0.xsd	includes fixml-orders-base-5-0.xsd Used to customize orders message category
Fixml-positions-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines positions messages: PositionMaintenanceRequest PositionMaintenanceReport RequestForPositions RequestForPositionsAck PositionReport AssignmentReport Adjusted Position Report ContraryIntentionReport
Fixml-positions-impl-5-0.xsd	Redefines fixml-positions-base-5-0.xsd Used to customize positions message category
Fixml-quotation-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines quotation messages: QuoteRequest Quote QuoteCancel QuoteStatusRequest MassQuoteAcknowledgement MassQuote QuoteRequestReject RFQRequest QuoteStatusReport QuoteResponse
Fixml-quotation-impl-5-0.xsd	Redefines fixml-quotation-base-5-0.xsd Used to customize quotations message category

Fixml-registration-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines registration messages: RegistrationInstructions RegistrationInstructionsResponse
Fixml-registration-impl-5-0.xsd	Includes fixml-registration-base-5-0.xsd Used to customize registration message category
Fixml-securitystatus-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines security status messages: SecurityDefinitionRequest SecurityDefinition SecurityStatusRequest SecurityStatus TradingSessionStatusRequest TradingSessionStatus SecurityTypeRequest SecurityTypes SecurityListRequest SecurityList DerivativeSecurityListRequest DerivativeSecurityList SecurityListUpdateReport SecurityDefinitionUpdateReport TradingSessionListRequest TradingSessionList
Fixml-securitystatus-impl-5-0.xsd	Includes fixml-securitystatus-base-5-0.xsd Used to customize security status message category
Fixml-settlement-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines settlement messages: SettlementInstructions SettlementInstructionsRequest
Fixml-settlement-impl-5-0.xsd	Includes fixml-settlement-base-5-0.xsd Used to customize settlement message category
Fixml-tradecapture-base-5-0.xsd	Includes fixml-components-impl-5-0.xsd . Defines trade capture messages: TradeCaptureReportRequest TradeCaptureReport TradeCaptureReportRequestAck TradeCaptureReportAck
Fixml-tradecapture-impl-5-0.xsd	Includes fixml-tradecapture-base-5-0.xsd Used to customize trade capture message category
Fixml-pretrade-5-0.xsd	Includes all pretrade message categories

Fixml-trade-5-0.xsd	Includes all trade message categories
Fixml-posttrade-5-0.xsd	Includes all post trade message categories
Fixml-main-5-0.xsd	Includes pretrade, trade, and posttrade schema files

COMMON COMPONENTS OF APPLICATION MESSAGES - Component Blocks **(Included in pre-trade, trade, and post-trade messages)**

Many of the FIX Application Messages are composed of common "building blocks" or sets of data fields. For instance, almost every FIX Application Message has the set of symbology-related fields used to define the "Instrument": Symbol, SymbolSfx, SecurityIDSource, SecurityID….. EncodedSecurityDesc. Rather than replicate a common group of fields, the FIX specification specifies several key component blocks below which are simply referenced by component name within each Application Message which uses them. Thus when reviewing a specific message definition, the appropriate group of fields should be expanded and used whenever a component block is identified.

Note that some component blocks may be part of repeating groups thus if the component block is denoted as part of a repeating group, then the entire group of fields representing the component block are to be specified at the component block's repeating group "level" in the message definition and follow repeating group rules concerning field order. See "Repeating Groups" for more details.

Instrument (symbology) component block

The Instrument component block contains all the fields commonly used to describe a security or instrument. Typically the data elements in this component block are considered the static data of a security, data that may be commonly found in a security master database. The Instrument component block can be used to describe any asset type supported by FIX.

Tag	*FieldName*	*Req'd*	*Comments*
55	Symbol	N	Common, "human understood" representation of the security. SecurityID value can be specified if no symbol exists (e.g. non-exchange traded Collective Investment Vehicles) Use "[N/A]" for products which do not have a symbol.
65	SymbolSfx	N	Used in Fixed Income with a value of "WI" to indicate "When Issued" for a security to be reissued under an old CUSIP or ISIN or with a value of "CD" to indicate a EUCP with lump-sum interest rather than discount price.
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.
454	NoSecurityAltID	N	
→	**455 SecurityAltID**	N	
→	**456 SecurityAltIDSource**	N	
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. Required for Fixed Income. Refer to Volume 7 - Fixed Income 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.)
762	SecuritySubType	N	Sub-type qualification/identification of the SecurityType (e.g. for SecurityType="MLEG"). If specified, SecurityType is required.
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.
1079	MaturityTime	N	
966	SettleOnOpenFlag	N	Indicator to determine if Instrument is Settle on Open.
1049	InstrmtAssignmentMethod	N	
965	SecurityStatus	N	Gives the current state of the instrument
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	(Deprecated in FIX.4.4)
226	RepurchaseTerm	N	(Deprecated in FIX.4.4)
227	RepurchaseRate	N	(Deprecated in FIX.4.4)
228	Factor	N	For Fixed Income: Amortization Factor for deriving Current face from Original face for ABS or MBS securities, note the fraction may be greater than, equal to or less than 1. In TIPS securities this is the Inflation index. Qty * Factor * Price = Gross Trade Amount For Derivatives: Contract Value Factor by which price must be adjusted to determine the true nominal value of one futures/options contract. (Qty * Price) * Factor = Nominal Value

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	(Deprecated in FIX.4.4)
202	StrikePrice	N	Used for derivatives, such as options and covered warrants
947	StrikeCurrency	N	Used for derivatives
967	StrikeMultiplier	N	Used for derivatives. Multiplier applied to the strike price for the purpose of calculating the settlement value.
968	StrikeValue	N	Used for derivatives. The number of shares/units for the financial instrument involved in the option trade.
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.
969	MinPriceIncrement	N	Minimum price increment for the instrument. Could also be used to represent tick value.
996	UnitofMeasure	N	Used to indicate the size of the underlying commodity on which the contract is based (e.g., 2500 lbs of lean cattle, 1000 barrels of crude oil, 1000 bushels of corn, etc.)
997	TimeUnit	N	Used to indicate a time unit for the contract (e.g., days, weeks, months, etc.)
223	CouponRate	N	For Fixed Income.
207	SecurityExchange	N	Can be used to identify the security.
970	PositionLimit	N	Position Limit for the instrument.
971	NTPositionLimit	N	Near-term Position Limit for the instrument.
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.
691	Pool		N	Identifies MBS / ABS pool
667	ContractSettlMonth		N	Must be present for MBS/TBA
875	CPProgram		N	The program under which a commercial paper is issued
876	CPRegType		N	The registration type of a commercial paper issuance
864	NoEvents		N	
→	**865**	**EventType**	N	
→	**866**	**EventDate**	N	
→	**867**	**EventPx**	N	
→	**868**	**EventText**	N	
873	DatedDate		N	If different from IssueDate
874	InterestAccrualDate		N	If different from IssueDate and DatedDate
component block <InstrumentParties>			N	Used to identify the parties listing a specific instrument

*** = Required status should match "Req'd" setting for <Instrument> component block in the message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML Element Instrmt

Examples using Alternative Security IDs

The SecurityAltID repeating group is used to carry additional security identifiers for the same security. Note that this repeating group can only be used inconjunction with the information in SecurityID and SecurityIDSource fields. In other words, it may not be used instead of the SecurityID and SecurityIDSource fields.

The first example is from an order for shares in Daimler Chrysler, which has an ISIN DE0007100000, a CUSIP D1668R123, and a Sedol 5529027

Field (tag)	Value	Explanation
Symbol (55)	DCX	Symbol = DCX (Daimler Chrysler)
SecurityID (48)	DE0007100000	
SecurityIDSource (22)	4	ID Type is ISIN
NoSecurityAltID (454)	2	Two additional security IDs specified
→ ***SecurityAltID (455)***	D1668R123	
→ ***SecurityAltIDSource (456)***	1	SecurityID type is Cusip
→ ***SecurityAltID (455)***	5529027	
→ ***SecurityAltIDSource (456)***	2	SecurityID type is Sedol

The second example is from an order for shares in IBM, which has an ISIN US4592001014, and a QUICK (Japanese) code of 000006680

Field (tag)	Value	Explanation
Symbol (55)	IBM	Symbol = IBM (International Business Machines)
SecurityID (48)	US4592001014	
SecurityIDSource (22)	4	ID Type is ISIN
NoSecurityAltID (454)	1	One additional security ID specified
→ ***SecurityAltID (455)***	000006680	
→ ***SecurityAltIDSource (456)***	3	SecurityID type is Quick

Specifying an FpML product specification from within the FIX Instrument Block

There are two methods in which a FpML product specification or document can be referenced from the FIX Instrument component block. The first method allows the full FpML product document to be embedded within the Instrument component block's EncodedSecurityDesc (350) field. The second method allows the FpML production document to be referenced as a URL in the Instrument component block. The tables below illustrates these two methods.

Option 1 – Include the FpML product specification as an XML String within EncodedSecurityDesc

Field (tag)	Value	Explanation
Symbol (55)	[N/A]	
SecurityID (48)	[FpML]	Refer to EncodedSecurityDesc for the FpML product decription,
SecurityIDSource (22)	I	ISDA/FpML Product Specification
EncodedSecurityDescLen (350)	1234	The length of the FpML product specification contained within EncodedSecurityDesc
EncodedSecurityDesc (351)	<FpML>….</FpML>	Contains the FpML product specification as an XML string

Option 2 – Reference the FpML product specification from another source via a URL in SecurityID

Field (tag)	Value	Explanation
Symbol (55)	[N/A]	
SecurityID (48)	(a valid URL reference)	Specify a URL to reference a separate or external location for the FpML product description. Example: ***http://www.cme.com/product/irswap.jpg?id=122345***
SecurityIDSource (22)	K	ISDA/FpML Product URL

UnderlyingInstrument (underlying instrument) component block

The UnderlyingInstrument component block, like the Instrument component block, contains all the fields commonly used to describe a security or instrument. In the case of the UnderlyingInstrument component block it describes an instrument which underlies the primary instrument ***Refer to the Instrument component block comments as this component block mirrors Instrument, except for the noted fields.***

Tag	*FieldName*	*Req'd*	*Comments*
311	UnderlyingSymbol	N	
312	UnderlyingSymbolSfx	N	
309	UnderlyingSecurityID	N	
305	UnderlyingSecurityIDSource	N	
457	NoUnderlyingSecurityAltID	N	
→	**458 UnderlyingSecurityAltID**	N	
→	**459 UnderlyingSecurityAltIDSource**	N	
462	UnderlyingProduct	N	
463	UnderlyingCFICode	N	
310	UnderlyingSecurityType	N	
763	UnderlyingSecuritySubType	N	
313	UnderlyingMaturityMonthYear	N	
542	UnderlyingMaturityDate	N	
241	UnderlyingCouponPaymentDate	N	
242	UnderlyingIssueDate	N	
243	UnderlyingRepoCollateralSecurityType	N	(Deprecated in FIX.4.4)
244	UnderlyingRepurchaseTerm	N	(Deprecated in FIX.4.4)
245	UnderlyingRepurchaseRate	N	(Deprecated in FIX.4.4)
246	UnderlyingFactor	N	
256	UnderlyingCreditRating	N	
595	UnderlyingInstrRegistry	N	
592	UnderlyingCountryOfIssue	N	
593	UnderlyingStateOrProvinceOfIssue	N	
594	UnderlyingLocaleOfIssue	N	
247	UnderlyingRedemptionDate	N	(Deprecated in FIX.4.4)

316	UnderlyingStrikePrice	N	
941	UnderlyingStrikeCurrency	N	
317	UnderlyingOptAttribute	N	
436	UnderlyingContractMultiplier	N	
998	UnderlyingUnitofMeasure	N	Used to indicate the size of the underlying commodity on which the contract is based (e.g., 2500 lbs of lean cattle, 1000 barrels of crude oil, 1000 bushels of corn, etc.)
1000	UnderlyingTimeUnit	N	Used to indicate a time unit for the contract (e.g., days, weeks, months, etc.)
435	UnderlyingCouponRate	N	
308	UnderlyingSecurityExchange	N	
306	UnderlyingIssuer	N	
362	EncodedUnderlyingIssuerLen	N	
363	EncodedUnderlyingIssuer	N	
307	UnderlyingSecurityDesc	N	
364	EncodedUnderlyingSecurityDescLen	N	
365	EncodedUnderlyingSecurityDesc	N	
877	UnderlyingCPProgram	N	
878	UnderlyingCPRegType	N	
972	UnderlyingAllocationPercent	N	Specific to the < UnderlyingInstrument > Percent of the Strike Price that this underlying represents. Necessary for derivatives that deliver into more than one underlying instrument.
318	UnderlyingCurrency	N	Specific to the <UnderlyingInstrument> (not in <Instrument>)
879	UnderlyingQty	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) Unit amount of the underlying security (par, shares, currency, etc.)
975	UnderlyingSettlementType	N	Specific to the < UnderlyingInstrument > Indicates order settlement period for the underlying deliverable component.
973	UnderlyingCashAmount	N	Specific to the < UnderlyingInstrument > Cash amount associated with the underlying component. Necessary for derivatives that deliver into more than one underlying instrument and one of the underlying's is a fixed cash value.
974	UnderlyingCashType	N	Specific to the < UnderlyingInstrument > Used for derivatives that deliver into cash underlying. Indicates

			that the cash is either fixed or difference value (difference between strike and current underlying price)
810	UnderlyingPx	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) In a financing deal clean price (percent-of-par or per unit) of the underlying security or basket.
882	UnderlyingDirtyPrice	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) In a financing deal price (percent-of-par or per unit) of the underlying security or basket. "Dirty" means it includes accrued interest
883	UnderlyingEndPrice	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) In a financing deal price (percent-of-par or per unit) of the underlying security or basket at the end of the agreement.
884	UnderlyingStartValue	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) Currency value attributed to this collateral at the start of the agreement
885	UnderlyingCurrentValue	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) Currency value currently attributed to this collateral
886	UnderlyingEndValue	N	Specific to the <UnderlyingInstrument> (not in <Instrument>) Currency value attributed to this collateral at the end of the agreement
component block <UnderlyingStipulations>		N	Specific to the <UnderlyingInstrument> (not in <Instrument>) Insert here the contents of the <UnderlyingStipulations> Component Block
1044	UnderlyingAdjustedQuantity	N	Specific to the <UnderlyingInstrument> (not in <Instrument>). For listed derivatives margin management, this is the number of shares adjusted for upcoming corporate action. Used only for securities which are optionable and are between ex-date and settlement date (4 days).
1045	UnderlyingFXRate	N	Specific to the <UnderlyingInstrument> (not in <Instrument>). Foreign exchange rate used to compute UnderlyingCurrentValue (885) (or market value) from UnderlyingCurrency (318) to Currency (15).
1046	UnderlyingFXRateCalc	N	Specific to the <UnderlyingInstrument> (not in <Instrument>). Specified whether UnderlyingFxRate (1045) should be multiplied or divided to derive UnderlyingCurrentValue (885).

1038	UnderlyingCapValue	N	
component block <UndlyInstrumentParties>		N	
1039	UnderlyingSettlMethod	N	

*** = Required status should match "Req'd" setting for <UnderlyingInstrument> component block in the message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element UndInstrmt

InstrumentLeg (symbology) component block

The InstrumentLeg component block, like the Instrument component block, contains all the fields commonly used to describe a security or instrument. In the case of the InstrumentLeg component block it describes a security used in multileg-oriented messages.

Refer to the Instrument component block comments as this component block mirrors Instrument, except for the noted fields.

Several multileg-oriented messages specify an Instrument Leg component block. An instrument can have zero or more instrument legs. The fundamental business rule that applies to the multileg instrument is that the multileg instrument is defined as the combination of instrument legs. The multileg instrument must be able to be traded atomically – that all instrument legs are traded or none are traded.

The LegRatioQty[623] is used to define the quantity of the leg that makes up a single unit of the multleg instrument. An option butterfly strategy is made up of three option legs.

Tag	*FieldName*	*Req'd*	*Comments*
600	LegSymbol	N	
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	
764	LegSecuritySubType	N	
610	LegMaturityMonthYear	N	
611	LegMaturityDate	N	
248	LegCouponPaymentDate	N	
249	LegIssueDate	N	
250	LegRepoCollateralSecurityType	N	(Deprecated in FIX.4.4)
251	LegRepurchaseTerm	N	(Deprecated in FIX.4.4)
252	LegRepurchaseRate	N	(Deprecated in FIX.4.4)
253	LegFactor	N	
257	LegCreditRating	N	
599	LegInstrRegistry	N	
596	LegCountryOfIssue	N	
597	LegStateOrProvinceOfIssue	N	

598	LegLocaleOfIssue	N	
254	LegRedemptionDate	N	(Deprecated in FIX.4.4)
612	LegStrikePrice	N	
942	LegStrikeCurrency	N	
613	LegOptAttribute	N	
614	LegContractMultiplier	N	
999	LegUnitofMeasure	N	Used to indicate the size of the underlying commodity on which the contract is based (e.g., 2500 lbs of lean cattle, 1000 barrels of crude oil, 1000 bushels of corn, etc.)
1001	LegTimeUnit	N	Used to indicate a time unit for the contract (e.g., days, weeks, months, etc.)
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>)
556	LegCurrency	N	Specific to the <InstrumentLeg> (not in <Instrument>)
740	LegPool	N	Identifies MBS / ABS pool
739	LegDatedDate	N	
955	LegContractSettlMonth	N	
956	LegInterestAccrualDate	N	

*** = Required status should match "Req'd" setting for <InstrumentLeg> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element InstrmtLeg

InstrumentExtension component block

The InstrumentExtension component block identifies additional security attributes that are more commonly found for Fixed Income securities.

Tag	*FieldName*	*Req'd*	*Comments*
668	DeliveryForm	N	Identifies the form of delivery.
869	PctAtRisk	N	Percent at risk due to lowest possible call.
870	NoInstrAttrib	N	
→	**871 InstrAttribType**	N	
→	**872 InstrAttribValue**	N	

*** = Required status should match "Req'd" setting for <InstrumentExtension> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element InstrmtExtension

OrderQtyData component block

The OrderQtyData component block contains the fields commonly used for indicating the amount or quantity of an order. Note that when this component block is marked as "required" in a message either one of these three fields must be used to identify the amount: OrderQty, CashOrderQty or OrderPercent (in the case of CIV).

Tag	*FieldName*	*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

*** = Required status should match "Req'd" setting for <OrderQtyData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element OrdQtyData

CommissionData component block

The CommissionDate component block is used to carry commission information such as the type of commission and the rate.

Tag	*FieldName*	*Req'd*	*Comments*
12	Commission	N	
13	CommType	N	
479	CommCurrency	N	For CIV - Optional
497	FundRenewWaiv	N	For CIV - Optional

*** = Required status should match "Req'd" setting for <CommissionData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML Element CommData

Parties component block

The Parties component block is used to identify and convey information on the entities both central and peripheral to the financial transaction represented by the FIX message containing the Parties Block. The Parties block allows many different types of entites to be expressed through use of the PartyRole field and identifies the source of the PartyID through the the PartyIDSource.

See “Volume 6 - APPENDIX 6-G - USE OF <PARTIES> COMPONENT BLOCK” for additional usage information..

Tag	*FieldName*		*Req'd*	*Comments*
453	NoPartyIDs		N	Repeating group below should contain unique combinations of PartyID, PartyIDSource, and PartyRole
→	**448**	**PartyID**	N	Used to identify source of PartyID. Required if PartyIDSource is specified. Required if NoPartyIDs > 0.
→	**447**	**PartyIDSource**	N	Used to identify class source of PartyID value (e.g. BIC). Required if PartyID is specified. Required if NoPartyIDs > 0.
→	**452**	**PartyRole**	N	Identifies the type of PartyID (e.g. Executing Broker). Required if NoPartyIDs > 0.
→	**802**	**NoPartySubIDs**	N	
→	→	**523** **PartySubID**	N	
→	→	**803** **PartySubIDType**	N	

*** = Required status should match "Req'd" setting for <Parties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element Ptys

NestedParties component block

The NestedParties component block is identical to the Parties Block. It is used in other component blocks and repeating groups when nesting will take place resulting in multiple occurrences of the Parties block within a single FIX message.. Use of NestedParties under these conditions avoids multiple references to the Parties block within the same message which is not allowed in FIX tag/value syntax.

Tag	*FieldName*	*Req'd*	*Comments*
539	NoNestedPartyIDs	N	Repeating group below should contain unique combinations of NestedPartyID, NestedPartyIDSource, and NestedPartyRole
→	**524 NestedPartyID**	N	Used to identify source of NestedPartyID. Required if NestedPartyIDSource is specified. Required if NoNestedPartyIDs > 0.
→	**525 NestedPartyIDSource**	N	Used to identify class source of NestedPartyID value (e.g. BIC). Required if NestedPartyID is specified. Required if NoNestedPartyIDs > 0.
→	**538 NestedPartyRole**	N	Identifies the type of NestedPartyID (e.g. Executing Broker). Required if NoNestedPartyIDs > 0.
→	**804 NoNestedPartySubIDs**	N	
→	→ **545 NestedPartySubID**	N	
→	→ **805 NestedPartySubIDType**	N	

*** = Required status should match "Req'd" setting for <NestedParties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element NstPtys

NestedParties2 (second instance of nesting) component block

The NestedParties2 component block is identical to the Parties Block. It is used in other component blocks and repeating groups when nesting will take place resulting in multiple occurrences of the Parties block within a single FIX message.. Use of NestedParties2 under these conditions avoids multiple references to the Parties block within the same message which is not allowed in FIX tag/value syntax.

Tag	*FieldName*		*Req'd*	*Comments*
756	NoNested2PartyIDs		N	Repeating group below should contain unique combinations of Nested2PartyID, Nested2PartyIDSource, and Nested2PartyRole
→	**757**	**Nested2PartyID**	N	Used to identify source of Nested2PartyID. Required if Nested2PartyIDSource is specified. Required if NoNested2PartyIDs > 0.
→	**758**	**Nested2PartyIDSource**	N	Used to identify class source of Nested2PartyID value (e.g. BIC). Required if Nested2PartyID is specified. Required if NoNested2PartyIDs > 0.
→	**759**	**Nested2PartyRole**	N	Identifies the type of Nested2PartyID (e.g. Executing Broker). Required if NoNested2PartyIDs > 0.
→	**806**	**NoNested2PartySubIDs**	N	
→	→	**760** **Nested2PartySubID**	N	
→	→	**807** **Nested2PartySubIDType**	N	

*** = Required status should match "Req'd" setting for <NestedParties2> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element NstPtys2

NestedParties3 (third instance of nesting) component block

The NestedParties3 component block is identical to the Parties Block. It is used in other component blocks and repeating groups when nesting will take place resulting in multiple occurrences of the Parties block within a single FIX message.. Use of NestedParties3 under these conditions avoids multiple references to the Parties block within the same message which is not allowed in FIX tag/value syntax.

Tag	*FieldName*		*Req'd*	*Comments*
948	NoNested3PartyIDs		N	Repeating group below should contain unique combinations of Nested3PartyID, Nested3PartyIDSource, and Nested3PartyRole
→	**949**	**Nested3PartyID**	N	Used to identify source of Nested3PartyID. Required if Nested3PartyIDSource is specified. Required if NoNested3PartyIDs > 0.
→	**950**	**Nested3PartyIDSource**	N	Used to identify class source of Nested3PartyID value (e.g. BIC). Required if Nested3PartyID is specified. Required if NoNested3PartyIDs > 0.
→	**951**	**Nested3PartyRole**	N	Identifies the type of Nested3PartyID (e.g. Executing Broker). Required if NoNested3PartyIDs > 0.
→	**952**	**NoNested3PartySubIDs**	N	
→	→	**953** **Nested3PartySubID**	N	
→	→	**954** **Nested3PartySubIDType**	N	

*** = Required status should match "Req'd" setting for <NestedParties3> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element NstPtys3

SettlParties (settlement parties) component block

The SettlParties component block is used in a similar manner as Parties Block within the context of settlement instruction messages to distinguish between parties involved in the settlement and parties who are expected to execute the settlement process.

Tag	*FieldName*		*Req'd*	*Comments*
781	NoSettlPartyIDs		N	Repeating group below should contain unique combinations of SettlPartyID, SettlPartyIDSource, and SettlPartyRole
→	**782**	**SettlPartyID**	N	Used to identify source of SettlPartyID. Required if SettlPartyIDSource is specified. Required if NoSettlPartyIDs > 0.
→	**783**	**SettlPartyIDSource**	N	Used to identify class source of SettlPartyID value (e.g. BIC). Required if SettlPartyID is specified. Required if NoSettlPartyIDs > 0.
→	**784**	**SettlPartyRole**	N	Identifies the type of SettlPartyID (e.g. Executing Broker). Required if NoSettlPartyIDs > 0.
→	**801**	**NoSettlPartySubIDs**	N	
→	→	**785** **SettlPartySubID**	N	
→	→	**786** **SettlPartySubIDType**	N	

*** = Required status should match "Req'd" setting for <SettlParties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element SettlPtys

SpreadOrBenchmarkCurveData component block

The SpreadOrBenchmarkCurveData component block is primarily used for Fixed Income to convey spread to a benchmark security or curve.

Tag	*FieldName*	*Req'd*	*Comments*
218	Spread	N	For Fixed Income
220	BenchmarkCurveCurrency	N	
221	BenchmarkCurveName	N	
222	BenchmarkCurvePoint	N	
662	BenchmarkPrice	N	
663	BenchmarkPriceType	N	Must be present if BenchmarkPrice is used.
699	BenchmarkSecurityID	N	The identifier of the benchmark security, e.g. Treasury against Corporate bond.
761	BenchmarkSecurityIDSource	N	Source of BenchmarkSecurityID. If not specified, then ID Source is understood to be the same as that in the Instrument block.

*** = Required status should match "Req'd" setting for <SpreadOrBenchmarkCurveData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element SpreadOrBnchmkCrvData

LegBenchmarkCurveData component block

The LegBenchmarkCurveData is used to convey the benchmark information used for pricing in a multi-legged Fixed Income security.

Tag	*FieldName*	*Req'd*	*Comments*
676	LegBenchmarkCurveCurrency	N	
677	LegBenchmarkCurveName	N	
678	LegBenchmarkCurvePoint	N	
679	LegBenchmarkPrice	N	
680	LegBenchmarkPriceType	N	

*** = Required status should match "Req'd" setting for <LegBenchmarkCurveData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element LegBnchmkCrvData

Stipulations component block

The Stipulations component block is used in Fixed Income to provide additional information on a given security. These additional information are usually not considered static data information.

Tag	*FieldName*		*Req'd*	*Comments*
232	NoStipulations		N	
→	**233**	**StipulationType**	N	Required if NoStipulations >0
→	**234**	**StipulationValue**	N	

*** = Required status should match "Req'd" setting for <Stipulations> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element Stips

UnderlyingStipulations component block

The UnderlyingStipulations component block has the same usage as the Stipulations component block, but for an underlying security.

Tag	*FieldName*		*Req'd*	*Comments*
887	NoUnderlyingStips		N	
→	**888**	**UnderlyingStipType**	N	Required if NoUnderlyingStips >0
→	**889**	**UnderlyingStipValue**	N	

*** = Required status should match "Req'd" setting for <UnderlyingStipulations> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element UndStips

LegStipulations component block

The LegStipulations component block has the same usage as the Stipulations component block, but for a leg instrument in a multi-legged security.

Tag	*FieldName*	*Req'd*	*Comments*
683	NoLegStipulations	N	
→	**688** **LegStipulationType**	N	Required if NoLegStipulations >0
→	**689** **LegStipulationValue**	N	

*** = Required status should match "Req'd" setting for <LegStipulations> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element LegStips

YieldData component block

The YieldData component block conveys yield information for a given Fixed Income security.

Tag	*FieldName*	*Req'd*	*Comments*
235	YieldType	N	
236	Yield	N	
701	YieldCalcDate	N	
696	YieldRedemptionDate	N	
697	YieldRedemptionPrice	N	
698	YieldRedemptionPriceType	N	

*** = Required status should match "Req'd" setting for <YieldData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element YldData

PositionQty component block

The PositionQty component block specifies the various types of position quantity in the position life-cycle including start-of-day, intraday, trade, adjustments, and end-of-day position quantities. Quantities are expressed in terms of long and short quantities.

Tag	*FieldName*	*Req'd*	*Comments*
702	NoPositions	N	
→	**703 PosType**	N	Required if NoPositions > 1
→	**704 LongQty**	N	
→	**705 ShortQty**	N	
→	**706 PosQtyStatus**	N	
→	**976 QuantityDate**	N	Date associated with the quantity being reported
→	**component block <NestedParties>**	N	Optional repeating group - used to associate or distribute position to a specific party other than the party that currently owns the position.

*** = Required status should match "Req'd" setting for <PositionQty> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element PosQty

PositionAmountData component block

The PositionAmountData component block is used to report netted amounts associated with position quantities. In the listed derivatives market the amount is generally expressing a type of futures variation or option premium. In the equities market this may be the net pay or collect on a given position.

Tag	*FieldName*		*Req'd*	*Comments*
753	NoPosAmt		N	Number of Position Amount entries
→	**707**	**PosAmtType**	N	
→	**708**	**PosAmt**	N	
→	**1055**	**PositionCurrency**	N	

*** = Required status should match "Req'd" setting for <PositionAmountData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element PosAmtData

TrdRegTimestamps component block

The TrdRegTimestamps component block is used to express timestamps for an order or trade that are required by regulatory agencies These timesteamps are used to identify the timeframes for when an order or trade is received on the floor, received and executed by the broker, etc.

Tag	*FieldName*		*Req'd*	*Comments*
768	NoTrdRegTimestamps		N	
→	**769**	**TrdRegTimestamp**	N	Required if NoTrdRegTimestamps > 1
→	**770**	**TrdRegTimestampType**	N	Required if NoTrdRegTimestamps > 1
→	**771**	**TrdRegTimestampOrigin**	N	
→	**1033**	**DeskType**	N	Type of Trading desk
→	**1034**	**DeskTypeSource**	N	
→	**1035**	**DeskOrderHandlingInst**	N	

*** = Required status should match "Req'd" setting for <TrdRegTimestamps> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element TrdRegTmstampsGrp

SettlInstructionsData component block

The SettlInstructionsData component block is used to convey key information regarding standing settlement and delivery instructions. It also provides a reference to standing settlement details regarding the source, delivery instructions, and settlement parties

It is important to understand that Settlement Instructions convey standing (reference) data only – and is not used for settlement transactions which are currently outside the scope of the FIX Protocol..

See "Volume 6 - APPENDIX 6-H - USE OF <SETTLINSTRUCTIONS> COMPONENT BLOCK" for additional usage information.

Tag	*FieldName*	*Req'd*	*Comments*
172	SettlDeliveryType	N	Required if AllocSettlInstType = 1 or 2
169	StandInstDbType	N	Required if AllocSettlInstType = 3 (should not be populated otherwise)
170	StandInstDbName	N	Required if AllocSettlInstType = 3 (should not be populated otherwise)
171	StandInstDbID	N	Identifier used within the StandInstDbType Required if AllocSettlInstType = 3 (should not be populated otherwise)
85	NoDlvyInst	N	
→	**165 SettlInstSource**	N	
→	**787 DlvyInstType**	N	
→	**component block <SettlParties>**	N	

*** = Required status should match "Req'd" setting for <SettlInstructionsData> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to the FIXML element SettlInstrctnsData

PegInstructions component block

The Peg Instructions component block is used to tie the price of a security to a market event such as opening price, mid-price, best price. The Peg Instructions block may also be used to tie the price to the behavior of a related security.

Tag	*FieldName*	*Req'd*	*Comments*
211	PegOffsetValue	N	Amount (signed) added to the peg for a pegged order in the context of the PegOffsetType
1094	PegPriceType	N	Defines the type of peg.
835	PegMoveType	N	Describes whether peg is static/fixed or floats
836	PegOffsetType	N	Type of Peg Offset (e.g. price offset, tick offset etc)
837	PegLimitType	N	Specifies nature of resulting pegged price (e.g. or better limit, strict limit etc)
838	PegRoundDirection	N	If the calculated peg price is not a valid tick price, specifies how to round the price (e.g. be more or less aggressive)
840	PegScope	N	The scope of the "related to" price of the peg (e.g. local, global etc)
1096	PegSecurityIDSource	N	Required if PegSecurityID is specified.
1097	PegSecurityID	N	Requires PegSecurityIDSource if specified.
1098	PegSymbol	N	
1099	PegSecurityDesc	N	

*** = Required status should match "Req'd" setting for <PegInstructions> component block in message definition

Note that Pegged orders are specified by the use of OrdType (to denote that the order is a pegged order) and ExecInst (to specify what price the order is pegged to).

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to the FIXML element PegInstrctns

DiscretionInstructions component block

The presence of DiscretionInstructions component block on an order indicates that the trader wishes to display one price but will accept trades at another price.

Tag	*FieldName*	*Req'd*	*Comments*
388	DiscretionInst	N	What the discretionary price is related to (e.g. primary price, display price etc)
389	DiscretionOffsetValue	N	Amount (signed) added to the "related to" price specified via DiscretionInst, in the context of DiscretionOffsetType
841	DiscretionMoveType	N	Describes whether discretion price is static/fixed or floats
842	DiscretionOffsetType	N	Type of Discretion Offset (e.g. price offset, tick offset etc)
843	DiscretionLimitType	N	Specifies the nature of the resulting discretion price (e.g. or better limit, strict limit etc)
844	DiscretionRoundDirection	N	If the calculated discretion price is not a valid tick price, specifies how to round the price (e.g. to be more or less aggressive)
846	DiscretionScope	N	The scope of "related to" price of the discretion (e.g. local, global etc)

*** = Required status should match "Req'd" setting for <DiscretionInstructions> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to the FIXML element DsctnInstrctns

FinancingDetails component block

Component block is optionally used only for financing deals to identify the legal agreement under which the deal was made and other unique characteristics of the transaction. The AgreementDesc field refers to base standard documents such as MRA 1996 Repurchase Agreement, GMRA 2000 Bills Transaction (U.K.), MSLA 1993 Securities Loan – Amended 1998, for example.

Tag	*FieldName*	*Req'd*	*Comments*
913	AgreementDesc	N	The full name of the base standard agreement, annexes and amendments in place between the principals and applicable to this deal
914	AgreementID	N	A common reference to the applicable standing agreement between the principals
915	AgreementDate	N	A reference to the date the underlying agreement was executed.
918	AgreementCurrency	N	Currency of the underlying agreement.
788	TerminationType	N	For Repos the timing or method for terminating the agreement.
916	StartDate	N	Settlement date of the beginning of the deal
917	EndDate	N	Repayment / repurchase date
919	DeliveryType	N	Delivery or custody arrangement for the underlying securities
898	MarginRatio	N	Percentage of cash value that underlying security collateral must meet.

*** = Required status should match "Req'd" setting for <FinancingDetails> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to the FIXML element FinancingDetails

ExpirationQty component block

The ExpirationQty component block identified the expiration quantities and type of expiration.

Tag	*FieldName*		*Req'd*	*Comments*
981	NoExpiration		N	
→	**982**	**ExpType**	N	Required if NoExpiration > 1
→	**983**	**ExpQty**	N	

*** = Required status should match "Req'd" setting for <ExpirationQty> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element ExpirationQty

SideTrdRegTS component block

The SideTrdRegTS component block is used to convey regulatory timestamps associated with one side of a multi-sided trade event.

Tag	*FieldName*		*Req'd*	*Comments*
1016	NoSideTrdRegTS		N	
→	**1012**	**SideTrdRegTimestamp**	N	
→	**1013**	**SideTrdRegTimestampType**	N	
→	**1014**	**SideTrdRegTimestampSrc**	N	

*** = Required status should match "Req'd" setting for <SideTrdRegTS> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element SideTrdTegTS

InstrumentParties component block

The use of this component block is restricted to instrument definition only and is not permitted to contain transactional information. Only a specified subset of party roles will be supported within the InstrumentParty block.

Possible uses of this block include identifying Listing Source information; Clearing Org information; Parent and Capital Structure information for F/I and derivative instruments.

Tag	*FieldName*			*Req'd*	*Comments*
1018	NoInstrumentParties			N	Repeating group below should contain unique combinations of InstrumentPartyID, InstrumentPartyIDSource, and InstrumentPartyRole
→	**1019**	**InstrumentPartyID**		N	Used to identify party id related to instrument
→	**1050**	**InstrumentPartyIDSource**		N	Used to identify source of instrument party id
→	**1051**	**InstrumentPartyRole**		N	Used to identify the role of instrument party id
→	**1052**	**NoInstrumentPartySubIDs**		N	
→	→	**1053**	**InstrumentPartySubID**	N	
→	→	**1054**	**InstrumentPartySubIDType**	N	

*** = Required status should match "Req'd" setting for <InstrumentParties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element InstrumentParties

UnderlyingAmount component block

The UnderlyingAmount component block is used to supply the underlying amounts, dates, settlement status and method for derivative positions.

Tag	*FieldName*		*Req'd*	*Comments*
984	NoUnderlyingAmounts		N	
→	**985**	**UnderlyingPayAmount**	N	Amount to pay in order to receive the underlying instrument.
→	**986**	**UnderlyingCollectAmount**	N	Amount to collect in order to deliver the underlying instrument.
→	**987**	**UnderlyingSettlementDate**	N	Date the underlying instrument will settle. Used for derivatives that deliver into more than one underlying instrument. Settlement dates can vary across underlying instruments.
→	**988**	**UnderlyingSettlementStatus**	N	Settlement status of the underlying instrument. Used for derivatives that deliver into more than one underlying instrument. Settlement can be delayed for an underlying instrument.

*** = Required status should match "Req'd" setting for <UnderlyingAmount> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element UnderlyingAmount Grp

DisplayInstruction component block

The DisplayInstruction component block is used to convey instructions on how a reserved order is to be handled in terms of when and how much of the order quantity is to be displayed to the market.

Tag	*FieldName*	*Req'd*	*Comments*
1138	DisplayQty	N	
1082	SecondaryDisplayQty	N	
1083	DisplayWhen	N	
1084	DisplayMethod	N	
1085	DisplayLowQty	N	Required when DisplayMethod = 3
1086	DisplayHighQty	N	Required when DisplayMethod = 3
1087	DisplayMinIncr	N	Can be used to specify larger increments than the standard increment provided by the market. Optionally used when DisplayMethod = 3
1088	RefreshQty	N	Required when DisplayMethod = 2

*** = Required status should match "Req'd" setting for <DisplayInstruction> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element DisplayInstruction Grp

TriggeringInstruction component block

The TriggeringInstruction component block specifies the conditions under which an order will be triggered by related market events as well as the behavior of the order in the market once it is triggered. .

Tag	*FieldName*	*Req'd*	*Comments*
1100	TriggerType	N	Required if any other Triggering tags are specified.
1101	TriggerAction	N	
1102	TriggerPrice	N	Only relevant and required for TriggerAction = 1
1103	TriggerSymbol	N	Only relevant and required for TriggerAction = 1
1104	TriggerSecurityID	N	Requires TriggerSecurityIDSource if specified. Only relevant and required for TriggerAction = 1
1105	TriggerSecurityIDSource	N	Requires TriggerSecurityIDSource if specified. Only relevant and required for TriggerAction = 1
1106	TriggerSecurityDesc	N	
1107	TriggerPriceType	N	Only relevant for TriggerAction = 1
1108	TriggerPriceTypeScope	N	Only relevant for TriggerAction = 1
1109	TriggerPriceDirection	N	Only relevant for TriggerAction = 1
1110	TriggerNewPrice	N	Should be specified if the order changes Price.
1111	TriggerOrderType	N	Should be specified if the order changes type.
1112	TriggerNewQty	N	Required if the order should change quantity
1113	TriggerTradingSessionID	N	Only relevant and required for TriggerType = 2.
1114	TriggerTradingSessionSubID	N	Requires TriggerTradingSessionID if specified. Relevant for TriggerType = 2 only.

*** = Required status should match "Req'd" setting for <TriggeringInstruction> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element TriggeringInstruction Grp

RootParties component block

The RootParties component block is a version of the Parties component block used to provide ***root information regarding*** the owning and entering parties of a transaction.

Tag	*FieldName*		*Req'd*	*Comments*
1116	NoRootPartyIDs		N	Repeating group below should contain unique combinations of RootPartyID, RootPartyIDSource, and RootPartyRole
→	**1117**	**RootPartyID**	N	Used to identify source of RootPartyID. Required if RootPartyIDSource is specified. Required if NoRootPartyIDs > 0.
→	**1118**	**RootPartyIDSource**	N	Used to identify class source of RootPartyID value (e.g. BIC). Required if RootPartyID is specified. Required if NoRootPartyIDs > 0.
→	**1119**	**RootPartyRole**	N	Identifies the type of RootPartyID (e.g. Executing Broker). Required if NoRootPartyIDs > 0.
→	**1120**	**NoRootPartySubIDs**	N	Repeating group of RootParty sub-identifiers.
→	→ **1121**	**RootPartySubID**	N	Sub-identifier (e.g. Clearing Acct for PartyID=Clearing Firm) if applicable. Required if NoRootPartySubIDs > 0.
→	→ **1122**	**RootPartySubIDType**	N	Type of Sub-identifier. Required if NoRootPartySubIDs > 0.

*** = Required status should match "Req'd" setting for <RootParties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element RootParties Grp

UndlyInstrumentParties component block

The use of this component block is restricted to instrument definition only and is not permitted to contain transactional information. Only a specified subset of party roles will be supported within the InstrumentParty block.

Possible uses of this block include identifying Listing Source information; Clearing Org information; Parent and Capital Structure information for F/Iixed Income and derivative instruments.

Tag	*FieldName*		*Req'd*	*Comments*
1058	NoUndlyInstrumentParties		N	Repeating group below should contain unique combinations of InstrumentPartyID, InstrumentPartyIDSource, and InstrumentPartyRole
→	**1059**	**UndlyInstrumentPartyID**	N	Used to identify party id related to instrument
→	**1060**	**UndlyInstrumentPartyIDSource**	N	Used to identify source of instrument party id
→	**1061**	**UndlyInstrumentPartyRole**	N	Used to identify the role of instrument party id
→	**1062**	**NoUndlyInstrumentPartySubIDs**	N	
→	→	**1063** **UndlyInstrumentPartySubID**	N	
→	→	**1064** **UndlyInstrumentPartySubIDType**	N	

*** = Required status should match "Req'd" setting for <UndlyInstrumentParties> component block in message definition

FIXML Definition for this Component Block– see http://www.fixprotocol.org for details
Refer to FIXML element UndlyInstrumentParties Grp

COMMON APPLICATION MESSAGES (Apply to pre-trade, trade, and post-trade)

Business Message Reject

The Business Message Reject message can reject an application-level message which fulfills session-level rules and cannot be rejected via any other means. Note if the message fails a session-level rule (e.g. body length is incorrect), a session-level Reject message should be issued.

See the session-level Reject message

It should ***NOT*** be used in the following situations:

Situation	Appropriate Response
Session-level problem meeting the criteria of the session-level Reject message	Use the session-level Reject message (MsgType=3)
In response to: • Quote Request	Use the Quote Request Reject message
In response to: • Quote • Quote Cancel • Quote Status Request • Quote Response	Use the Quote Status Report message
In response to: • Mass Quote	Use the Mass Quote Acknowledgment message
In response to: • Market Data Request	Use the Market Data Request Reject message
In response to: • Security Definition Request	Use the Security Definition message
In response to: • Security Type Request	Use the SecurityTypes message
In response to: • Security List Request	Use the Security List message
In response to: • Derivative Security List Request	Use the Derivative Security List message
In response to: • Security Status Request	Use the Security Status message
In response to: • Trading Session Status Request	Use the Trading Session Status message
In response to	Use the Execution Report message

• New Order - Single • Order Status Request • Order Mass Status Request • New Order – Cross • New Order – Multileg • New Order – List • List Execute	
In response to: • Order Cancel Request • Order Cancel/Replace Request • Cross Order Cancel Request • Cross Order Cancel/Replace Request • Mulileg Order Cancel/Replace Request • List Cancel Request	Use the Order Cancel Reject message
In response to: • Execution Report	Use the Don't Know Trade (DK) message or the Execution Report Acknowledgement message
In response to: • Order Mass Cancel Request	Use the Order Mass Cancel Report message
In response to: • List Status Request	Use the List Status message
In response to: • Allocation Instruction	Use the Allocation Instruction Ack message
In response to: • Allocation Report	Use the Allocation Report Ack message
In response to: • Confirmation	Use the Confirmation Ack message
In response to: • Registration Instructions	Use the Registration Instructions Response message
In response to: • Trade Capture Report Request	Use the Trade Capture Report message
In response to: • Bid Request	Use the Bid Response message
In response to: • Confirmation Request	Use the Confirmation message
In response to:	Use the Settlement Instructions message

• Settlement Instruction Request	
In response to: • Position Maintenance Request	Use the Position Maintenance Report message
In response to: • Request for Positions	Use the Request for Positions Ack message
In response to: • Collateral Request	Use the Collateral Assignment message
In response to: • Collateral Assignment	Use the Collateral Response message
In response to: • Collateral Inquiry	Use the Collateral Inquiry Ack message

Note the only exceptions to this rule are:

1. **in the event a business message is received, fulfills session-level rules, however, the message cannot be communicated to the business-level processing system.** In this situation a Business Message Reject with BusinessRejectReason = "Application not available at this time" can be issued if the system is unable to send the specific "reject" message listed above due to this condition.
2. **in the event a valid business message is received, fulfills session-level rules, however, the message type is not supported by the receipient.** In this situation a Business Message Reject with BusinessRejectReason = "Unsupported Message Type" can be issued if the system is unable to send the specific "reject" message listed above because the receiving system cannot generate the related "reject" message.
3. **In the event a business message is received, fulfills session-level rules, but lacks a field conditionally required by the FIX specification.** In this situation a Business Message Reject with BusinessRejectReason = "Conditionally Required Field Missing" can be issued if the system is unable to send the specific "reject" message listed above. One example of this would be a stop order missing StopPx. However, a Business Message Reject message **MUST NOT** be used to enforce proprietary rules more restrictive than those explicit in the FIX specification, such as a broker requiring an order to contain an Account, which the FIX specification considers an optional field.

Messages which can be referenced via the Business Message Reject message are:

(the "ID" field BusinessRejectRefID refers to noted in [ ])
- Indication of Interest (IOI) [IOIid]
- Advertisement [AdvId]
- News [Headline]
- Email [EmailThreadID]
- Order Cancel Reject [ClOrdID]
- Allocation Instruction ACK [AllocID]
- Allocation Report ACK [AllocID]
- List Status [ListID]
- Don't Know Trade (DK) – may respond with Order Cancel Reject if attempting to cancel order [ExecID]
- Settlement Instructions [SettlInstID]

- Market Data-Snapshot/Full Refresh [MDReqID]
- Market Data-Incremental Refresh [MDReqID]
- Market Data Request Reject [MDReqID]
- Execution Report Acknowledgement [ExecID]
- Security Definition [SecurityResponseID]
- Security Status [SecurityStatusReqID]
- Trading Session Status [TradSesReqID]
- Order Mass Cancel Report [OrderID]
- Security Types [SecurityResponseID]
- Security List [SecurityResponseID]
- Derivative Security List [SecurityResponseID]
- Quote Request Reject [QuoteReqID]
- RFQ Request [RFQReqID]
- Quote Status Report [QuoteID]
- Registration Instructions Response [RegistID]
- Trade Capture Report [TradeReportID]
- Confirmation ACK [ConfirmID]
- Bid Response [BidID]
- List Strike Price [ListID]
- Settlement Instructions [SettInstMsgID]
- Trade Capture Report Request Ack [TradeRequestID]
- Trade Capture Report Ack [TradeReportID]
- Position Maintenance Report [PosMaintRptID]
- Request for Positions Ack [PosMaintRptID]
- Positions Report [PosMaintRptID]
- Assignment Report [AsgnRptID]
- Collateral Response [CollRespID]
- Collateral Inquiry Ack [CollInquiryID]

Scenarios for Business Message Reject:

BusinessRejectReason
0 = Other
1 = Unkown ID
2 = Unknown Security
3 = Unsupported Message Type (receive a valid, but unsupported MsgType)
4 = Application not available
5 = Conditionally Required Field Missing

Whenever possible, it is strongly recommended that the cause of the failure be described in the Text field (e.g. "UNKNOWN SYBMOL: XYZ").

The business message reject format is as follows:

Business Message Reject

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = j (lowercase)
45	RefSeqNum	N	MsgSeqNum of rejected message
372	RefMsgType	Y	The MsgType of the FIX message being referenced.
379	BusinessRejectRefID	N	The value of the business-level "ID" field on the message being referenced. Required unless the corresponding ID field (see list above) was not specified.
380	BusinessRejectReason	Y	Code to identify reason for a Business Message Reject message.
58	Text	N	Where possible, message to explain reason for rejection
354	EncodedTextLen	N	Must be set if EncodedText field is specified and must immediately precede it.
355	EncodedText	N	Encoded (non-ASCII characters) representation of the Text field in the encoded format specified via the MessageEncoding field.
	Standard Trailer	Y	

FIXML Definition for this Message– see http://www.fixprotocol.org for details
Refer to the FIXML element BizMsgRej

Network Status Messages

It is envisaged these messages will be used in two scenarios:

Scenario A

Allow one counterparty using a "hub and spoke" FIX network to know whether another counterparty is currently connected to the hub (i.e. whether the counterparty's session to the hub is up or not).

Scenario B

Allow a counterparty connecting to a global brokerage to know which regions within that brokerage are currently available as order routing destinations.

Network (Counterparty System) Status Request Message

This message is send either immediately after logging on to inform a network (counterparty system) of the type of updates required or to at any other time in the FIX conversation to change the nature of the types of status updates required. It can also be used with a NetworkRequestType of Snapshot to request a one-off report of the status of a network (or counterparty) system. Finally this message can also be used to cancel a request to receive updates into the status of the counterparties on a network by sending a NetworkRequestStatusMessage with a NetworkRequestType of StopSubscribing.

Network (Counterparty System) Status Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = "BC"
935	NetworkRequestType	Y	
933	NetworkRequestID	Y	
936	NoCompIDs	N	Used to restrict updates/request to a list of specific CompID/SubID/LocationID/DeskID combinations. If not present request applies to all applicable available counterparties. EG Unless one sell side broker was a customer of another you would not expect to see information about other brokers, similarly one fund manager etc.
→	***930*** ***RefCompID***	N	Used to restrict updates/request to specific CompID
→	***931*** ***RefSubID***	N	Used to restrict updates/request to specific SubID
→	***283*** ***LocationID***	N	Used to restrict updates/request to specific LocationID
→	***284*** ***DeskID***	N	Used to restrict updates/request to specific DeskID
	Standard Trailer	Y	

FIXML Definition for this Message– see http://www.fixprotocol.org for details
Refer to the FIXML element NtwkSysStatReq

Network (Counterparty System) Status Response Message

This message is sent in response to a Network (Counterparty System) Status Request Message.

If the network response payload is larger than the maximum permitted message size for that FIX conversation the response would be several Network Status Response Messages the first with a status of full and then as many messages, as updates to the first message, adding information as required.

Network (Counterparty System) Status Response

Tag	Field Name		Req'd	Comments
	Standard Header		Y	MsgType = "BD"
937	NetworkStatusResponseType		Y	
933	NetworkRequestID		N	
932	NetworkResponseID		Y	
934	LastNetworkResponseID		N	Required when NetworkStatusResponseType=2
936	NoCompIDs		Y	Specifies the number of repeating CompId's
→	***930***	***RefCompID***	N	CompID that status is being report for. Required if NoCompIDs > 0,
→	***931***	***RefSubID***	N	SubID that status is being report for.
→	***283***	***LocationID***	N	LocationID that status is being report for.
→	***284***	***DeskID***	N	DeskID that status is being report for.
→	***928***	***StatusValue***	N	
→	***929***	***StatusText***	N	Additional Information, i.e. "National Holiday"
	Standard Trailer		Y	

FIXML Definition for this Message– see http://www.fixprotocol.org for details

Refer to the FIXML element NtwkSysStatRsp

User Administration Messages

These messages are provided in FIX to allow the passing of individual user information between two counterparties. The messages allow for the following function

1 – Individual User Logon

2 – Individual User Status Enquiries

3 – Individual User Logout

4 – Individual User password change

NOTE: While it is not encouraged to transmit passwords in a FIX conversation unless you can guarantee the end to end security of both the FIX conversation and any intermediate routing hubs that are involved in the routing.

User Request Message

This message is used to initiate a user action, logon, logout or password change. It can also be used to request a report on a user's status.

User Request

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = "BE"
923	UserRequestID	Y	
924	UserRequestType	Y	
553	Username	Y	
554	Password	N	
925	NewPassword	N	
95	RawDataLength	N	
96	RawData	N	Can be used to hand structures etc to other API's etc
	Standard Trailer	Y	

FIXML Definition for this Message– see http://www.fixprotocol.org for details
Refer to the FIXML element UserReq

User Response Message

This message is used to respond to a user request message, it reports the status of the user after the completion of any action requested in the user request message.

User Response

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = "BF"
923	UserRequestID	Y	
553	Username	Y	
926	UserStatus	N	
927	UserStatusText	N	Reason a request was not carried out
	Standard Trailer	Y	

FIXML Definition for this Message– see http://www.fixprotocol.org for details
Refer to the FIXML element UserRsp

Glossary

Business Terms

The following glossary is an attempt to identify business terms used in this document or related to implementing FIX globally. Requests for new terms and/or suggested definitions should be posted in the FIX Web Site's Discussion section.

Term	Definition	Field where used
Accrued Interest Rate	The amount the buyer compensates the seller for the portion of the next coupon interest payment the seller has earned but will not receive from the issuer because the issuer will send the next coupon payment to the buyer. Accrued Interest Rate is the annualized Accrued Interest amount divided by the purchase price of the bond.	
After Tax Yield	Municipals. The yield on the bond net of any tax consequences from holding the bond. The discount on municipal securities can be subject to both capital gains taxes and ordinary income taxes. Calculated from dollar price.	[YieldType]
All or None	A round-lot market or limit-price order that must be executed in its entirety or not at all; unlike Fill or Kill orders, AON orders are not treated as canceled if they are not executed as soon as represented in the Trading Crowd.	[ExecInst]
Annual Yield	The annual interest or dividend income an investment earns, expressed as a percentage of the investment's total value.	[YieldType]
As defined	Sides of the legs are the same as defined in the multileg instrument.	[Side]
At the close	Indicated price is to be around the closing price, however, not held to the closing price.	[IOIQualifier]
At the Opening	A market or limit-price order to be executed at the opening of the stock or not at all; all or part of any order not executed at the opening is treated as canceled.	[TimeInForce]
Basis Price	A price established by joint agreement of odd-lot dealers in 100-share-unit stocks when: - no round-lot has occurred during the trading session, - the spread between the closing bid and offer is two points or more, and - on odd-lot the dealer has been given a "basis-price" order.	[OrdType]
Book Yield	The yield of a security calculated by using its book value instead of the current market price. This term is typically used in the US domestic market.	[YieldType]
Broker Execution	According to US futures markets (CFTC): Time at which a broker executed the order for another broker.	[TrdRegTimestampType]
Broker of Credit	Broker to receive trade credit.	[PartyRole]
Broker Receipt	According to US futures markets (CFTC):	[TrdRegTimestampType]

	Time at which broker received the order.	
Buy Minus	A round-lot market order to buy “minus” is an order to buy a stated amount of a stock provided that its price is: - not higher than the last sale if the last sale was a “minus” or “zero minus” tick and - not higher than the last sale minus the minimum fractional change in the stock if the last sale was a “plus” or “zero plus” tick. A limit price order to buy “minus” also states the highest price at which it can be executed.	[Side]
Cabinet Trade	An off-market transaction to close out a nearly worthless out-of-the-money option contract.	
Call Date	The date on which the issuer of a security has the right but not the obligation to repurchase the security at a predetermined price.	[EventType]
Call First	Refer to client before trading.	[ExecInst]
Cancel if Not Best	Indicates that an order should be cancelled if it is no longer the best bid if buying, or the best offer if selling. If the order is cancelled due to this instruction, the message cancelling it must carry ExecRestatementReason="Canceled, Not Best".	[ExecInst]
Cancel on System Failure	If a system failure interrupts trading or order routing, attempt to cancel this order. Note that depending on the type and severity of the failure, this might not be possible.	[ExecInst]
Cancel on Trading Halt	If trading in this instrument is halted, cancel this order and do not reinstate it when/if trading resumes.	[ExecInst]
CIV ("Collective Investment Vehicle")	Collective investment vehicle ("CIV") are set up for the purposes of collecting and pooling investor funds and issuing shares (or their equivalent). "Open-ended" CIVs entitle the holder to receive, on demand, an amount in value which is proportionate to the whole net asset value of the vehicle. Coversely "Closed-ended" CIVs do not grant this right to investors. CIVs are more commonly known as Mutual Funds, Unit Trusts, OEICS (Open Ended Investment Companies), SICAVs etc. A CIV may be legally constituted as a Limited Company with variable capital, a Trust or a Limited Partnership - depending on local legislation & tax regimes. CIVs typically invest in equities, bonds, derivatives etc. - as described in their prospectus. Other CIVs are Umbrella Fund (made up of sub-funds investing in equities, gilts etc), Fund Of Funds (invests only in other funds), Master-Feeder Fund (marketed to a specific group for investment in a central master fund), Multi-Manager Fund (whose asset management is divied between several managers), Side By Side (onshore and offshore funds with the same investment strategy)	[CFICode] and a “Product” within Volume 7
Clearing Firm	Firm that will clear the trade. Used if different from the executing firm.	[PartyRole]
Clearing Organization	Identifies the Clearing Organization where the position is	[PartyRole]

	maintained.	
Client ID	Firm identifier used in third party-transactions or for investor identification in intermediary transactions. (should not be a substitute for OnBehalfOfCompID/DeliverToCompID).	[PartyRole]
Closing Yield	The yield of a bond based on the closing price.	[YieldType]
Closing Yield Most Recent Month	The yield of a bond based on the closing price as of the most recent month's end.	[YieldType]
Closing Yield Most Recent Quarter	The yield of a bond based on the closing price as of the most recent quarter's end.	[YieldType]
Closing Yield Most Recent Year	The yield of a bond based on the closing price as of the most recent year's end.	[YieldType]
Compound Yield	The yield of certain Japanese bonds based on its price. Certain Japanese bonds have irregular first or last coupons, and the yield is calculated compound for these irregular periods.	[YieldType]
Contra Firm	The broker or other firm which is the contra side of the trade.	[PartyRole]
Contra Clearing Firm	Clearing firm of the broker or other firm which is the contra side of the trade.	[PartyRole]
Contra Trader	Individual usually identified by a trading badge number or initials that takes the opposite side of a trade.	[PartyRole]
Contract For Difference (CFD)	A single stock total return swap, combining financing and synthetic equity exposure in one transaction.	[BookingType]
Correspondent Broker	Identifies a correspondent broker.	[PartyRole]
Correspondent Clearing Firm	ClearingFirm that is going to carry the position on their books at another clearing house (exchanges).	[PartyRole]
Correspondent Clearing Organization	Identifies a correspondent clearing organization	[PartyRole]
Coupon Rate	The rate of interest that, when multiplied by the principal, par value, or face value of a bond, provides the currency amount of the periodic interest payment. The coupon is always cited, along with maturity, in any quotation of a bond's price.	
Cross	Client sends Broker a buy or sell order. Broker wishes to take the other side and cross with the client. Broker sends an order with Side=Cross to an exchange.	[OrdType]
Cross Short	Client wants to establish a short position, and so sends a Sell Short to Broker. Broker wants to cross with the Client, so Broker sends a Cross Short order to an exchange. Cross Short is crucial here because many exchanges have tick rules needing to be enforced, and the order getting converted from Sell Short to Cross (instead of Cross Short) could result in an illegal short sell.	[OrdType]
Cross Short Exempt	Client wants to establish a short position, and is exempt from the uptick restriction. So Client sends Sell Short Exempt to Broker. Broker wants to cross with the Client, so Broker needs a way to send "Cross Short Exempt" to the exchange so that an audit trail traces back indicating that the party selling short was exempt from the uptick rule.	[OrdType]

Currency swap	"An agreement to exchange future cash flows. There are two fundamental types: the cross-currency swap and the interest rate (single currency) swap." *Source: A Foreign Exchange Primer by Shani Shamah*	
Current Yield	Annual interest on a bond divided by the market value. The actual income rate of return as opposed to the coupon rate expressed as a percentage.	[YieldType]
Customer Account	Identifies the customer account associated with the message.	[PartyRole]
Dated Date	The effective date of a new securities issue determined by its underwriters. Often but not always the same as the "Issue Date" and the "Interest Accrual Date"	
Day Order	A buy or sell order that, if not executed expires at the end of the trading day on which it was entered.	[TimeInForce]
Dealt currency	In a foreign exchange transaction, 'dealt currency' indicates which currency was orginally specified. For example, An invesment manager may 'buy 100M USD against EUR'. This has USD as the 'dealt currency' and EUR as the 'counter currency'. Note that when viewed from the sell-side's (or broker's) perspective, this is a 'Sell 100M USD against EUR' the 'dealt currency' remains the same.	[Currency]
Discount	When a bond sells below its par value, it is said to be selling at a discount. A price with a PriceType of "discount" is the difference between 100 and the bond's percent-of-par price.	[PriceType]
Do Not Increase	A limit order to buy, a stop order to sell, or a stop-limit order to sell which is not to be increased in shares on the ex-dividend date as a result of a stock dividend or distribution.	[ExecInst]
Do Not Reduce	A limit order to buy, a stop order to sell, or a stop-limit order to sell that is not to be reduced in price by the amount of an ordinary cash dividend on the ex-dividend date. A do-not-reduce order applies only to ordinary cash dividends; it should be reduced for other distributions - such as when a stock goes “ex” stock dividend or “ex” rights.	[ExecInst]
Dollar Price	See "Percent of Par"	[PriceType]
Entering Firm	Broker who has recorded or reported an execution. This field is particularly useful where the trade is entered into a trade recording system by a broker who is not a party to the trade, as it allows any inquiries or problem resolution to be directed to the appropriate source.	[PartyRole]
Entering Trader	Individual usually identified by a trading badge number or initials that actually enters an order to a market (especially in open outcry markets). Usually the Entering Trader is the same as the Executing Trader. However, under some circumstances the Entering Trader will have the trade executed by another trader who is then identified as the Executing Trader.	[PartyRole]
Even swap	An FX Swap where the given amount to be bought and sold is the same on the near and far legs. See "uneven swap".	
Exchange	Exchange associated with the position.	[PartyRole]

Execution Time	According to US futures markets (CFTC): Non-qualified reporting time of order execution.	[TrdRegTimestampType]
Executing Firm	Identifies executing / give-up broker.	[PartyRole]
Executing System	System Identifier where execution took place (some markets have multiple execution location such as an electronic book or automatic execution system).	[PartyRole]
Executing Trader	Trader or broker id associated with Executing Firm who actually executes the trade.	[PartyRole]
External Routing Allowed	Indicates that an order sent to one market may be routed by that market to other external markets, especially in cases where the order locks or crosses the market and it can be executed against another market's superior price. Note: The absence of this instruction does not imply that an order should not be routed externally; rather, the order receiver's default will apply.	[ExecInst]
External Routing Not Allowed	Indicates that an order sent to one market may never be routed by that market to other external markets. Should the order lock or cross the market but be unable to execute due to price protection reasons, a market may have to take alternate action, which might include rejecting the order, depending on the market's rules. Note: The absence of this instruction does not imply that an order should be routed externally; rather, the order receiver's default will apply.	[ExecInst]
Fill or Kill	A market or limit-price order that is to be executed in its entirety as soon as it is represented in the Trading Crowd; if not so executed, the order is to be canceled. Not to be confused with Immediate or Cancel.	[TimeInForce]
FIX Connection	**A FIX Connection is comprised of three parts: logon, message exchange, and logout.**	
FIX Session	**A FIX Session is comprised of one or more FIX Connections, meaning that a FIX Session spans multiple logins.**	
Fixed Price Cabinet Trade	Cabinet Trade executed at a price equal to the minimum tick size (or smallest possible price) . See "Cabinet Trade".	[PriceType]
Floating Price Cabinet Trade	Cabinet Trade executed at a price that can be different than the minimal price. See "Cabinet Trade".	[PriceType]
Forex - Swap	A "Swap" order for Foreign Exchange (currency trading).	[OrdType]
Foreign exchange swap	The transaction of exchanging two currencies at an agreed upon rate at an agreed upon time. The transaction is reversed at a future rate and time, with no payment streams between the points in time. *Source: A paraphrase of definition from* *http://joxo.co.uk/SummaryGuideToFXForFinancialandITProfessionals.html*	
Funari	Japanese term for an order to buy or sell a stated amount of a security at the specified limit price with any unexecuted (leftover) quantity becoming a Market On Close order.	[OrdType]

Fund manager Client ID	***For CIV:*** An identifier for an Investor or a broker or funds supermarket's nominee/custodian company which is recognized by the Fund manager.	[PartyRole]
Giveup Clearing Firm	Firm to which the trade is given up (carries the position that results from a trade).	[PartyRole]
Good Till Canceled	An order to buy or sell that remains in effect until it is either executed or canceled; sometimes called an "open order".	[TimeInForce]
Government Equivalent Yield	Ask yield based on semi-annual coupons compounding in all periods and actual/actual calendar.	[YieldType]
Held	The firm executing the order is held to best execution requirements, and may not make discretionary decisions. Opposite of Not Held	[ExecInst]
Ignore Price Validity Checks	Disables validity checking of price fields for an order or change request.	[ExecInst]
Immediate or Cancel	A market or limit-price order that is to be executed in whole or in part as soon as it is represented in the Trading Crowd; any portion not so executed is to be canceled. Not to be confused with Fill or Kill.	[TimeInForce]
Initiator	An "initiator" may be one of the following: • an institutional client • a financial planner • a retail broker representing a retail customer • a broker/dealer • an inter-dealer broker (or broker's broker) • an issuer	Quoting and other messages Volume 7
Institutions Only	Broker is restricted to dealing with other buy side firms.	[ExecInst]
Interest Accrual Date	The start date used for calculating accrued interest on debt instruments which are being sold between interest payment dates. Often but not always the same as the "Issue Date" and the "Dated Date".	
Intermarket sweep	An order that is an intermarket sweep as defined by the SEC in Regulation NMS. This value is used on Immediate or Cancel limit orders (or other order type and time in force). It indicates that the party sending the order has taken responsibility for price protection, and the recipient of the order should execute it, if possible, without regard to protection of other markets' prices. While the term "Intermarket sweep" is specific to the United States, the ExecInst value that represents it may be used in other markets, where appropriate, to indicate an order that should be executed without regard to price protection.	[ExecInst]
Introducing Firm	The broker or other intermediary with the closest association with the investor.	[PartyRole]
Inverse Floater Bond Yield	Inverse floater semi-annual bond equivalent rate.	[YieldType]
Investor ID	***For Equities:*** Identifies beneficiary or broker acting on behalf of beneficiary. This field is mandatory for various exchanges either pre or post	[PartyRole]

	trade. Numerical entry containing no dashes.	
	For CIV: An Investor identifier such as a taxpayer reference (NINO, NPN, DSS, SSN number etc) for an individual investor or a registration number (EIN, etc.) for a company. May contain alphanumeric and dashes.	[PartyRole]
Issue Date	The date on which a bond or stock offering is issued. It may or may not be the same as the effective date ("Dated Date") or the date on which interest begins to accrue ("Interest Accrual Date")	
Limit	An order to buy a security at or below a stated price, or to sell a security at or above a stated price.	[OrdType]
Limit or Better	Indicates an order to - buy a security at the indicated limit price or lower, or to - sell a security at the indicated limit price or higher.	[OrdType]
Limit With or Without	An order to be executed at a limit price, with or without round-lot sales; valid only for odd lot orders.	[OrdType]
Liquidity Provider	Identifies the individual that provided liquidity, e.g. was the market maker (specialist) involved in a trade. Used to identify the liquidity provider involved in a block of EFP trade for listed futures markets.	[PartyRole]
Locate/Lending Firm	Identity of the firm which is loaning the security in a short sale.	[PartyRole]
Marked To Market Yield	An adjustment in the valuation of a securities portfolio to reflect the current market values of the respective securities in the portfolio.	[YieldType]
Market	Indicates an order to buy or sell a stated amount of a security at the most advantageous price obtainable after the order is represented in the Trading Crowd.	[OrdType]
Market If Touched	Indicates an order to buy or sell a stated amount of a security or commodity as soon as a preset market price is reached, at which point it becomes a Market order.	[OrdType]
Market On Close	Indicated price is held to the closing price ("firm" instruction).	[IOIQualifier]
Market Or Better	Indicates an order to buy or sell a stated amount of a security at the quoted market or better.	[OrdType]
Market with Leftover as Limit	Indicates an order to buy or sell a stated amount of a security at the prevailing market price with any unexecuted (leftover) quantity becoming a Limit order at the last executed price.	[OrdType]
Most Recent Closing Yield	The last available yield stored in history, computed using price.	[YieldType]
Next Fund Valuation Point	For CIV orders - indicates that the Investor wishes the order to be dealt at the unit price determined at the next Valuation Point, a.k.a. a Forward price.	[OrdType]
No Cross	The broker executing this trade is forbidden from taking the other side of the trade. Opposite of OK to Cross.	[ExecInst]

Not Held	The firm executing the order is not held to best execution requirements, and may be able to make some discretionary decisions. Opposite of Held.	[ExecInst]
OK to Cross	The broker executing this trade is allowed to take the other side of the trade. Opposite of No Cross.	[ExecInst]
Omnibus Account	An account where the positions for multiple entities (usually customers) are maintained. The omnibus accounting is usually done on a gross basis where long and short positions are not netted together.	[PartyRole]
On Basis	An order to buy or sell at the basis price. The basis price is established by joint agreement of odd lot dealers in 100 share unit stocks when no round lot sale has occurred during the trading session, the spread between the closing bid and offer is two points or more, and an odd lot dealer has been given a basis price order.	[OrdType]
Opposite	Sides of the legs are the opposite of their definition in the multileg instrument.	[Side]
Order Origination Firm	Buyside firm associated with Order Origination Firm which originates/submits the order.	[PartyRole]
Order Origination Trader	Buyside trader id associated with Order Origination Firm which originates/submits the order.	[PartyRole]
Par	Equal to the face value (nominal) of a security, e.g. A bond selling at par is worth an equivalent to its original issue value, typically $1000/bond.	[QuantityType]
Participate Don't Initiate	An order that may participate in a transaction initiated by another party, but may not initiate a transaction. For example, on US ECNs / Exchanges, this may represent an order that will be quoted to the marketplace and will trade if another party initiates a trade (i.e. hits the posted quote), but cannot be routed to initiate a trade with another market or market maker.	[ExecInst]
Per Unit	The currency price per unit, e.g. per equity share or per contract.	[PriceType]
Percent of Par	The ratio between the current price of a bond and its par value adjusted for amortization or indexing and expressed as a percent. For example if a EUR1,000 bond is trading at EUR1032.50 its price is expressed as 103.25 or 103¼. In the US this is usually referred to as the "dollar price" even in scholarly material and handbooks.	[PriceType]
Position Account	Account for positions resulting from derivatives trades. Each position account has a long and short quantity. Position quantities stored in the long and short quantity fields can be kept net or gross. Accounts that are kept gross are usually omnibus accounts.	[PartyRole]
Percent of Volume	The sender does not want to be all of the volume on the floor.	[ExecInst]
Premium	When a bond sells above its par value, it is said to be selling at a premium. A price with a PriceType of "premium" is the difference between the bond's percent-of-par price and 100.	[PriceType]
Previous Fund Valuation Point	For CIV orders - indicates that the Investor prefers that the order be dealt at the unit price determined at the immediately preceding	[OrdType]

	Valuation Point, a.k.a. a Historic price. (This can be overridden by the constitution of the fund or, in certain circumstances, by the Fund Manager.)	
Open Average Yield	The average yield of the respective securities in the portfolio.	[YieldType]
Order Originator	ID of the party entering the trade into the system (data entry, userid, buy side trader, etc.).	[PartyRole]
Put Date	The date on which the buyer of a security has the right but not the obligation to sell the security back to the issuer at a predetermined price.	[EventType]
Previous Close Yield	The yield of a bond based on the closing price 1 day ago.	[YieldType]
Previously indicated	An order sent in response to an Indication of Interest message.	[OrdType]
Previously quoted	An order sent in response to a Quote message.	[OrdType]
Proceeds Yield	The CD equivalent yield when the remaining time to maturity is less than two years.	[YieldType]
Redeem	***For CIV:*** A "sell" order for CIV units which must be forwarded to the fund manager (or their transfer agent) rather than being matched / crossed with a "buy" order, e.g. by an intermediary, funds supermarket, broker/dealer etc. This would be used in markets where the originator requires specific tax treatment and/or dealing charges.	[Side]
Reinstate on System Failure	If a system failure interrupts trading or order routing, attempt to reinstate this order, subject to time in force limitations. Note that depending on the type and severity of the failure, this might not be possible.	[ExecInst]
Reinstate on Trading Halt	If trading in this instrument is halted, reinstate this order when/if trading resumes, subject to time in force limitations.	[ExecInst]
Request to Intermediary	Used in a model where an intermediary, e.g. clearing house, is involved in communicating allocation details and actions between two parties	[AllocType]
Respondent	A "respondent" may be one of the following: • a broker/dealer • an inter-dealer broker (or broker's broker) • an electronic service • bid or offer prices provided by one or more market makers • bid or offer prices provided by an inter-dealer broker • matching system with limit orders entered by customers (dealers or institutions) • an issuer	Quoting and other messages Volume 7
Riskless Principal	"Riskless" principal transactions are generally described as trades in which, after receiving an order to buy (or sell) from a customer, the broker-dealer purchases (or sells) the security from (or to) another person in a contemporaneous offsetting transaction. Above from the SEC web-site ***http://www.sec.gov/rules/final/34-44291.htm***	[OrderCapacity]

	See Exchange Act Rule 10b-10(a)(2)(ii)(A) [17 CFR 240. 10b-10(a)(2)(ii)(A)]; Exchange Act Rel. No. 33743 (Mar. 9, 1994) at n.11.	
Sell Plus	A round-lot market order to sell “plus” is an order to sell a stated amount of a stock provided that its price is: - not lower than the last sale if the last sale was a “plus” or “zero plus” tick and - not lower than the last sale minus the minimum fractional change in the stock if the last sale was a “minus” or “zero minus” tick. A limit-price order to sell “plus” also states the lowest price at which it can be executed.	[OrdType]
Sell Short	An order to sell a security that the seller does not own; a sale effected by delivering a security borrowed by, or for the account of, the seller. Can only be executed on a “plus” or “zero plus” tick.	[OrdType]
Sell Short Exempt	Short sale exempt from short-sale rules.	[OrdType]
Semi-annual Yield	The yield of a bond whose coupon payments are reinvested semi-annually.	[YieldType]
Settlement currency	Commonly referred to as "counter currency" in FX nomenclature. For non-NDF deals (FX swaps, spot and forward) the term "settlement currency" can only mean one thing: the currency that is on the opposite from the dealt currency (expressed in FIX using Currency field (tag 15)). For example: Symbol is EUR/USD, and the dealt is EUR then SettlCurrency is USD. For NDF deals the term "settlement currency" could be either the dealt currency or the "counter currency" or a third currency. For example: In a USD/KRW NDF deal where the dealt currency is KRW, the settlement currency is USD, if the dealt currency is USD then the settlement currency can also be USD. In a GBP/KRW NDF deal where the deal typically settles in a third currency, USD in this case, then the settlement currency is USD. (NDFs will be discussed in detail in Phase 2 gap analysis). For FX OTC Spot Options, the settlement currency can refer to either the counter currency or the currency of the option premium (or premia). However, for the purposes of FIX usage, it will refer to the currency of the option premium.	
Settlement Location	Identifies Settlement Depository or, if local settlement, the ISO Country Code.	[PartyRole]
Sponsoring Firm	A member of the exchange that is sponsoring an Entering Entity to send orders to the exchange. The Sponsoring Member Firm permits sponsorees (e.g. Entering Entities) to trade thereby allowing them to enter orders directly to the exchange via automated means. (e.g. NYSE allowing direct access via Anonymous DOT service).	[PartyRole]
Spread	A "spread" price is one of four things all denominated in basis points: 1) For an outright security trade, the "spread" price is the difference in yield between the object security and its	[PriceType]

	benchmark - implied or explicit. 2) For a swap (or switch) of two issued securities the "spread" price is the difference in yield between the security being sold and the one being bought. 3) For a roll of a futures contract with a contract in the same commodity but having a different contract settlement month the "spread" price is the price difference between the contract being sold and the one being bought. 4) For a floating-rate Financing transaction the “spread” is the difference in yield extended above or below the yield of the stated benchmark. All four types are expressed in basis points (the price or yield difference times 100) and may be negative.	
Stop	A stop order to buy which becomes a market order when the security trades at - or above - the stop price after the order is represented in the Trading Crowd. A stop order to sell which becomes a market order when the security trades at - or below - the stop price after the order is represented in the Trading Crowd.	[OrdType]
Stop Limit	A stop order to buy which becomes a limit order at the limit price when the security trades at - or above - the stop price after the order is represented in the Trading Crowd. A stop order to sell which becomes a limit order at the limit price when the security trades at - or below- the stop price after the order is represented in the Trading Crowd.	[OrdType]
Stopped	A trade is guaranteed for the order, usually at a stated price or better, but has not yet occurred. For example, a specialist on an exchange may "stop" an order while searching for a better price.	[OrdStatus]
Streetside Trade Capture Reporting	Reporting of completed trades for clearance and settlement or compliance purposes. Reports may be originated by Exchanges or by clearing firms and sent to clearing firms directly or via a clearing corporation or central counterparty such as DTCC in the US.	A “Section” in “Volume 5”
Simple Yield	The yield of a bond assuming no reinvestment of coupon payments. (Act/360 day count)	[YieldType]
Strict Limit (No Price Improvement)	A limit order that must be traded at the exact limit price specified without any price improvement. Requires OrdType=Limit.	[ExecInst]
Subscribe	***For CIV:*** A “buy” order for CIV units which must be forwarded to the fund manager (or their transfer agent) rather than being matched / crossed with a “sell” order, e.g. by an intermediary funds supermarket, broker/dealer etc. This would be used in markets where the originator requires specific tax treatment and/or dealing charges.	[Side]
Suspended	The order is not eligible for trading. This usually happens as a result of a verbal or otherwise out of band request to suspend the order, or because the order was submitted, or modified via a Cancel/Replace Request, with ExecInst=Suspended.	[OrdStatus]

Tax Equivalent Yield	The after tax yield grossed up by the maximum federal tax rate of 39.6%. For comparison to taxable yields.	[YieldType]
TED Price	The price spread between the active 3 month treasury bill futures contract and the 3 month Eurodollar futures contract. Used as an indicator of investor confidence in the U.S. markets.	[PriceType]
TED Yield	The difference in basis points between the yield-to-maturity of the bond / note and the yield-to-maturity of a Hypothetical Euromarket bond with identical coupon and maturity.	[PriceType]
Time In	According to US futures markets (CFTC): Timestamp of when order was received on the trading floor (booth).	[TrdRegTimestampType]
Time Out	According to US futures markets (CFTC): Timestamp when the trade was received from the pit.	[TrdRegTimestampType]
Trade Along	Clients who specify "Trade Along" give brokers permission to handle and place their order in the market even if the broker already has its own proprietary orders for the same security placed in the market.	[ExecInst]
Trailing Stop Peg	A pegged order representing a stop order with a stop price pegged to trail a specified distance behind the last sale price. The price of a trailing stop to buy can never increase, and the price of a trailing stop to sell can never decrease.	[ExecInst]
True Gross Yield	Yield calculated using the price including accrued interest, where coupon dates are moved from holidays and weekends to the next trading day.	[YieldType]
True Yield	The yield calculated with coupon dates moved from a weekend or holiday to the next valid settlement date.	[YieldType]
Try to Stop	Used in specialist-driven markets to direct the specialist to try and stop the order.	[ExecInst]
Underlying Contra Firm	The broker or other firm which is the contra side of the trade for the underlying security.	[PartyRole]
Uneven swap	An FX Swap where the given amount to be bought and sold is different on the near and far legs.	
URI (Uniform Resource Identifier)	W3C standard defined as "the generic set of all names/addresses that are short strings that refer to resources". Note that "URL" (Uniform Resource Locator), commonly referred to by web browsers, is a subset of the URI standard. The W3C standards body considers URL an "informal term (no longer used in technical specifications)".	See Appendix 6-B
Value date	The delivery or settlement date of a foreign exchange trade transaction.	
With or Without	An odd lot order filled on an effective round lot transaction, or on an effective bid or offer, whichever occurs first after the specialist receives the order. (e.g. NYSE order type)	[OrdType]
Yield At Issue	Municipals. The yield of the bond offered on the issue date.	[YieldType]

Yield Change Since Close	The change in the yield since the previous day's closing yield.	[YieldType]
Yield To Average Maturity	The yield achieved by substituting a bond's average maturity for the issue's final maturity date.	[YieldType]
Yield To Next Call	The yield of a bond to the next possible call date.	[YieldType]
Yield To Longest Average Life	The yield assuming only mandatory sinks are taken. This results in a lower paydown of debt; the yield is then calculated to the final payment date.	[YieldType]
Yield To Maturity	The yield of a bond to its maturity date.	[YieldType]
Yield To Next Put	The yield to the date at which the bond holder can next put the bond to the issuer.	[YieldType]
Yield To Next Refund	Sinking Fund Bonds. Yield assuming all bonds are redeemed at the next refund date at the redemption price.	[YieldType]
Yield To Shortest Average Life	The yield assuming that all sinks (mandatory and voluntary) are taken at par. This results in a faster paydown of debt; the yield is then calculated to the final payment date.	[YieldType]
Yield To Tender Date	The yield on a Municipal bond to its mandatory tender date.	[YieldType]
Yield To Worst	The lowest yield to all possible redemption date scenarios.	[YieldType]
Yield Value of 1/32	The amount that the yield will change for a 1/32nd change in price.	[YieldType]
Yield with Inflation Assumption	Based on price, the return an investor would require on a normal bond that would make the real return equal to that of the inflation-indexed bond, assuming a constant inflation rate.	[YieldType]

Appendix 1-A: Abbreviations used within FIXML

Acrl	Accrual
Acct	Account
Ack	Acknowledgement
Acrd	Accrued
Actn	Action
Adj	Adjust
Adjmt	Adjustment
Adv	Advertisement
Alloc	Allocation
Amt	Amount
AOS	AllowableOneSidedness
Asgn	Assignment
Avg	Average
Bhf	Behalf
Bkng	Booking
Bnchmk	Benchmark
Brkr	Broker
Brkrs	Brokers
Biz	Business
Calc	Calculation
Capt	Capture
Ccy	Currency
Cl	Client
Cls	Close
Cmn	Common
Cnfm	Confirmation Confirm
Cntra	Contra
Coll	Collateral
Comm	Commission
Comp	Company
Corp	Corporate
Cpcty	Capacity

Cpn	Coupon
Crss	Cross
Crv	Curve
Csh	Cash
Ctry	Country
Cum	Cumulative
Cxl	Cancel
Data	Data
Db	Database
Del	Delete
Desc	Description
Dest	Destination
Dev	Device
Disc	Discount
DK	Don't Know
Dlvr	Deliver
Dsctn	Discretion
Dsctnry	Discretionary
Dt	Date
Dup	Duplicate
Efctv	Effective
EFP	ExchangeForPhysical
Enc	Encoded
Err	Error
Exct	Execute
Exch	Exchange
Exctn	Execution
Exr	Exercise
Fctr	Factor
Fut	Future
Fwd	Forward
FX	Foreign Currency
Grp	Group

GTD	Good Till Date
Hndl	Handling
ID	Identifier
Implct	Implicit
Ind	Indicator
Info	Information
Inpt	Input
Inq	Inquiry
Instrctn	Instruction
Instn	Institution
Instrmt	Instrument
Int	Interest
IOI	Indication of Interest
Iss	Issue
Issr	Issuer
Lctn	Location
Loc	Locate
Lqdty	Liquidity
Mat	Maturity
Max	Maximum
Mgn	Margin
Min	Minimum
Mkt	Market
Mleg	Multileg
Mnt	Maintenance
Mny	Money
Mo	Month
Mod	Modification
Misc	Miscellaneous
Msg	Message
Mtch	Match
Ndx	Index
No	Number - NumInGroup fields
Nst	Nested
Ntwk	Network

Num	Number - multiple reports, counts
Ofr	Offer
Opt	Option
Ord	Order
Orig	Original
Oth	Other
Pct	Percent
Pctg	Percentage
Pmt	Payment
Pos	Position
Prod	Product
Pri	Priority
Prlm	Preliminary
Prtztn	Priotization
Prev	Previous
Psbl	Possible
Pty	Party
Pub	Publish
Px	Price
Qlty	Quality
Qty	Quantity
Qual	Qualifier
Quot	Quote
Red	Redemption
Ref	Reference
Rej	Reject
Reltd	Related
Repo	Repurchase
Req	Request
Rgst	Registration
Rgstry	Registry
Rnd	Round
Rpt	Report
Rpts	Reports

Rslt	Result
Rsn	Reason
Rsp	Response
Rstct	Restrict
Rstctn	Restriction
Rstctns	Restrictions
Rstmt	Restatement
Rt	Rate
Rtng	Rating
Scnd	Secondary
Sec	Security
Seq	Sequence
Sess	Session
Settl	Settlement
Sfx	Suffix
Shrt	Short
Snd	Sender
Sndg	Sending
Src	Source
St	State
Stand	Standing
Stat	Status
Stip	Stipulation
Strk	Strike
Sub	Subscription

Subsid	Subsidiary
Svc	Service
Sym	Symbol
Sys	System
Sz	Size
Tgt	Target
Tkt	Ticket
Tm	Time
Tot	Total
Typ	Type
Trd	Trade
TrdSes	TradingSession
Trkng	Tracking
Trm	Term
TS	Timestamp
Txn	Transaction
Und	Underlying
Valu	Value
Vol	Volume
Yld	Yield
Yr	Year

