High Performance Working Group

Simple Binary Encoding – Gap Analysis

March 26, 2013

For Global Technical Committee Governance Internal Use Only

Submission Date	March 7, 2013	Control Number	
Submission Status	Submitted	Ratified Date	
Primary Contact Person	Fred Malarbre, CME Group	Release Identifier	

r3

Revision v0.5

Proposal Status: Submitted

For Global Technical Committee Governance Internal Use Only

Submission Date	March 7, 2013	Control Number	
Submission Status	Submitted	Ratified Date	
Primary Contact Person	Fred Malarbre, CME Group	Release Identifier	

DISCLAIMER

Table of Contents

Document History

Revision	Date	Author	Revision Comments
v0.1	2013-02-14	Fred Malabre	Initial draft
v0.2	2013-02-21	Fred Malabre	Included feedback from Hanno
v0.3	2013-02-22	Fred Malabre	Additional updates based on HPWG review
v0.4	2013-02-25	Fred Malabre	Add an example for inner groups encoding to appendix D.
v0.5	2013-03-26	Don Mendelson	Binary message definitions changed to complete schema, formatting, minor corrections

1 Introduction

1.1 Background

The High Performance Working Group was formed with the goal of improving the fit-for-purposefulness of FIX for high performance.

Recent improvements in the speed of hardware, software, and network connections (such as in co-location solutions) are putting pressure on the FIX protocol and highlighting some inefficiencies of the current version of the protocol (e.g., excessive echoing of input values, inefficient encoding). New financial applications such as high-frequency trading and market data feeds pose new performance requirements. In recent years, several financial organizations have avoided the performance limitations of FIX and introduced new proprietary protocols that are optimized for speed. These proprietary interfaces have been offered, sometimes along with a FIX interface, to support high-speed transactions and/or data feeds.

The current performance limitations of FIX can be removed by making changes and additions at multiple levels of the protocol. At the *application* level, there is a need to define less-verbose versions of some FIX messages and to streamline the message flow. At the *presentation* level, there is a need to provide new encodings that are faster and more compact than the traditional Tag=Value encoding of FIX. At the *session* level, there is a need to specify a new lightweight session protocol with basic recovery options. The High Performance Working Group is drafting a set of specifications and guideline documents to address all these aspects.

This proposal entails the use of an FPL designed *Simple Binary Encoding* to produce fast and compact encodings of FIX messages.

Simple Binary Encoding provides different characteristics than other binary encodings. It is optimized for low latency. This new FPL binary encoding complements the existing only binary encoding developed in 2005 (FAST) with a focus on reducing bandwidth utilization for market data. In addition, the encoding is also defined and controlled within FPL only in contrast to the binary encodings proposals to encode FIX with Google Protocol Buffers and ASN.1.

2 Business Workflow

Simple Binary Encoding supports a full mapping of the existing FIX specification to efficient binary messages. This new encoding is intended to join the ranks of existing FIX encodings—ASCII Tag=Value, FIXML, and FAST, to provide an array of encoding options to support the varying needs of different organizations, while preserving the semantic richness of the FIX interface.

Simple Binary Encoding will be subject to versioning, this proposal covers Version 1.0 of this encoding. Optimization will be provided in subsequent versions as discussed in the next chapter.

Its definition is comprised of a message schema describing the encoded fields as well as a encoded messages containing only the values to be sent.

Rules on message schema construction are very relaxed to allow the flexibility to optimize a specific implementation for high performance. This includes the choice of big-endian or little-endian field encoding, positions of the fields within a message independent of the FIX layout, constant length encoded fields provided to have full direct access to any content within a message. In addition, support for variable length strings and n-deep repeating groups is also provided with a trade off for direct access. This flexibility allows implementations to focus on different aspects of their business to support non-optimized legacy messages mixed with optimized messages as defined within the Application Layer Subgroup of the High Performance Working Group.

It is proposed that the attached technical specification draft be admitted into the FPL standardization process and eventually be made available to FIX implementers and users.

2.1 Encoding principles

2.1.1 Binary type system

In order to support traditional FIX semantics, all the documented field types are supported. However, instead of printable character representations of tag-value encoding, the type system binds to native binary data types, and defines derived types as needed.

The binary type system has been enhanced in these ways:

- Provides a means to specify precision of decimal numbers and timestamps, as well as valid ranges of numbers.
- Differentiates fixed-length character arrays from variable-length strings. Allows a way to specify the minimum and maximum length of strings that an application can accept.
- Provides a consistent system of enumerations, Boolean switches and multiple-choice fields.

2.1.2 Design principles

The message design strives for direct data access without complex transformations or conditional logic. This is achieved by:

- Usage of native binary data types and simple types derived from native binaries, such as prices and timestamps.
- Preference for fixed positions and fixed length fields, supporting direct access to data and avoiding the need for management of heaps of variable-length elements which must be sequentially processed.

2.1.3 Message schema

This standard describes how fields are encoded and the general structure of messages. The content of a message type is specified by a message schema. A message schema tells which fields belong to a message and their location within a message. Additionally, the metadata describes valid value ranges and information that need not be sent on the wire, such as constant values.

Message schemas may be based on standard FIX message specifications, or may be customized as needed by agreement between counterparties.

2.2 Field Encoding

2.2.1 Field aspects

A field is a unit of data contained by a FIX message. Every field has the following aspects: semantic data type, encoding, and metadata. They will be specified in more detail in the sections are data type encoding and message schema but are introduced here as an overview.

2.2.2 Semantic data type

The FIX semantic data type of a field tells a data domain in a broad sense, for example, whether it is numeric or character data, or whether it represents a time or price. Simple Binary Encoding represents all of the semantic data types that FIX protocol has defined across all encodings. In message specifications, FIX data type is declared with attribute fixUsage. See the Simple Binary Encoding technical specification for a listing of those FIX types.

2.2.3 Encoding

Encoding tells how a field of a specific data type is encoded on the wire. An encoding maps a FIX data type to either a simple, primitive data type, such as a 32 bit signed integer, or to a composite type. A composite type is composed of two or more simple types. For example, the FIX data type Price is encoded as a decimal, a composite type containing a mantissa and an exponent. Note that many fields may share a data type and an encoding. The sections that follow explain the valid encodings for each data type.

3 Issues and Discussion Points

This proposal contains the "core" features for the Simple Binary Encoding (Version 1.0).

Extensions will be proposed to enhance Simple Binary Encoding to handle more complex use cases. Below are examples of extensions being studied:

- Handle specific use cases related to repeating groups encodings that can be optimized to allow direct access to inner groups as well as direct processing without parsing parent level groups. Designing these extensions will require extensive testing on specific use cases and thus will be proposed as an extension of the "core" proposal.
- Handle self describing messages consisting of the encoded wire message prepended with an encoded template would support minimal processing for archiving data encoded with Simple Binary Encoding independently of external templates.

4 Proposed Message Flow

This proposal does not include any new message flow.

5 FIX Message Tables

This section does not apply to this proposal.

6 FIX Component Blocks

This section does not apply to this proposal.

7 Category Changes

This section does not apply to this proposal.

Appendix A - Data Dictionary

This section does not apply to this proposal.

Appendix B - Glossary Entries

This section does not apply to this proposal.

Appendix C - Abbreviations

This section does not apply to this proposal.

Appendix D - Usage Examples

D1 – No Repeating Groups – NewOrderSingle Message

D1-1 – Plain English Representation

Entry of buy limit order for a quantity of 10 for instrument 5764 at the price of 96.25.

D1-2 – FIX tag=value Encoding Representation (MsgType=”D”)

```
8=FIXT.1.1|9=121|35=D|49=TA|56=EA|34=12|52=20130222-
13:34:32.128|11=Order56|48=5764|22=8|54=1|60=20130222-
13:34:32.109|38=10|40=2|44=96.25|10=087|
```

D1-3 – Simple Binary Encoding Message Schema

```
<sbe:messageSchema xmlns:sbe="http://www.fixprotocol.org/ns/simple/1.0"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
        <types>
                <!-- Simple types -->
                <type name="octet" primitiveType="uint8" />
                <type name="unsignedShort" primitiveType="uint16" />
                <type name="unsigned" primitiveType="uint32" />
                <type name="unsignedLong" primitiveType="uint64" />

                <!-- Constants -->
                <type name="FIXT11" primitiveType="char" presence="constant">FIXT.1.1</type>
                <type name="msgTypeD" primitiveType="char" presence="constant">D</type>
                <type name="sender" primitiveType="char" length="2" presence="constant">TA</type>
                <type name="target" primitiveType="char" length="2" presence="constant">EA</type>
                <type name="ExchangeSymbol" primitiveType="char" presence="constant">8</type>

                <!-- Composite types -->
                <composite name="decimal32"
                         description="decimal constant exponent; only sends mantissa on the wire">
                        <type name="mantissa" primitiveType="int32" />
                        <type name="exponent" primitiveType="int8" presence="constant">-2</type>
                </composite>
                <composite name="qty32"
                        description="32 bit quantity constrained to integer values">
                        <type name="mantissa" primitiveType="int32" />
                        <type name="exponent" primitiveType="int8" presence="constant">0</type>
                </composite>

                <!-- Enumerations -->
                <enum name="OrdTypeEnum" encodingType="char">
                        <validValue name="Market">1</validValue>
                        <validValue name="Limit">2</validValue>
                        <validValue name="Stop">3</validValue>
                        <validValue name="StopLimit">4</validValue>
                        <validValue name="MarketOnClose">5</validValue>
                        <validValue name="WithOrWithout">6</validValue>
                        <validValue name="LimitOrBetter">7</validValue>
                        <validValue name="LimitWithOrWithout">8</validValue>
                        <validValue name="OnBasis">9</validValue>
                        <validValue name="OnClose">A</validValue>
                        <validValue name="LimitOnClose">B</validValue>
                        <validValue name="ForexMarket">C</validValue>
                        <validValue name="PreviouslyQuoted">D</validValue>
                        <validValue name="PreviouslyIndicated">E</validValue>
                        <validValue name="ForexLimit">F</validValue>
                        <validValue name="ForexSwap">G</validValue>
                        <validValue name="ForexPreviouslyQuoted">H</validValue>
                        <validValue name="Funari">I</validValue>
                        <validValue name="MarketIfTouched">J</validValue>
                        <validValue name="MarketWithLeftOverAsLimit">K</validValue>
                        <validValue name="PreviousFundValuationPoint">L</validValue>
                        <validValue name="NextFundValuationPoint">M</validValue>
                        <validValue name="Pegged">P</validValue>
                        <validValue name="CounterOrderSelection">Q</validValue>
                </enum>

                <enum name="SideEnum" encodingType="char">
                        <validValue name="Buy">1</validValue>
                        <validValue name="Sell">2</validValue>
                        <validValue name="BuyMinus">3</validValue>
                        <validValue name="SellPlus">4</validValue>
                        <validValue name="SellShort">5</validValue>
                        <validValue name="SellShortExempt">6</validValue>
                        <validValue name="Undisclosed">7</validValue>
                        <validValue name="Cross">8</validValue>
                        <validValue name="CrossShort">9</validValue>
                        <validValue name="CrossShortExempt">A</validValue>
                        <validValue name="AsDefined">B</validValue>
                        <validValue name="Opposite">C</validValue>
                        <validValue name="Subscribe">D</validValue>
                        <validValue name="Redeem">E</validValue>
                        <validValue name="Lend">F</validValue>
                        <validValue name="Borrow">G</validValue>
                </enum>

        </types>

        <!-- Message definition -->
        <sbe:message name="SimpleNewOrderSingle" id="1" fixMsgType="D"
                description="Simplified NewOrderSingle">
                <field name="BeginString" id="8" type="FIXT11" fixUsage="String" />
                <field name="BodyLength" id="9" type="octet" fixUsage="Length" />
                <field name="MsgType" id="35" type="msgTypeD" fixUsage="String" />
                <field name="SenderCompID" id="49" type="sender" fixUsage="String" />
                <field name="TargetCompID" id="56" type="target" fixUsage="String" />
                <field name="MsgSeqNum" id="34" type="unsigned" fixUsage="SeqNum" />
                <!-- Using midnight as epoch instead of UNIX epoch, which is default -->
                <field name="SendingTime" id="52" type="unsignedLong"
                        timeUnit="millisecond" epoch="00:00" fixUsage="UTCTimestamp" />
```

```
            <field name="ClOrdID" id="11" type="string8" fixUsage="String" />
            <field name="SecurityID" id="48" type="unsignedShort" fixUsage="String" />
            <field name="SecurityIDSource" id="22" type="ExchangeSymbol" fixUsage="String" />
            <field name="Side" id="54" type="SideEnum" fixUsage="char" />
            <field name="TransactTime" id="60" type="unsignedLong"
                   timeUnit="millisecond" fixUsage="UTCTimestamp" />
            <field name="OrderQty" id="38" type="qty32" fixUsage="Qty" />
            <field name="OrdType" id="40" type="OrdTypeEnum" fixUsage="char" />
            <field name="Price" id="44" type="decimal32" fixUsage="Price" />
            <field name="CheckSum" id="10" type="octet" fixUsage="int"/>
        </sbe:message>

</sbe:messageSchema>
```

D1-4 – Simple Binary Encoding Wire Format

Annotated message

```
27                      | [H] EncodedLength = 39
01                      | [H] MessageID = 1
                        | [8] BeginString = FIXT.1.1 (constant)
                        | [9] BodyLength =
                        | [35] MsgType = D (constant)
                        | [49] SenderCompID = TA (constant)
                        | [56] TargetCompID = EA (constant)
00 00 00 0C             | [34] MsgSeqNum = 12
00 00 01 3D 02 1C C2 C0 | [52] SendingTime = 20130222-13:34:32.128
4F 72 64 65 72 35 36 20 | [11] ClOrdID = Order56
16 84                   | [48] SecurityID = 5764
                        | [22] SecurityIDSource = 8 (constant)
01                      | [54] Side = 1
00 00 01 3D 02 1C C2 AD | [60] TransactTime = 20130222-13:34:32.109
0A                      | [38] OrderQty = 10
02                      | [40] OrdType = 2
                        | [44] Price
00 00 25 99             |       mantissa = 9625
                        |       exponent = -2 (constant)
57                      | [10] CheckSum
```

Raw message

```
27 01 00 00 00 0C 00 00 01 3D 02 1C C2 C0 4F 72 64 65 72 35 36 20 16 84 01 00
00 01 3D 02 1C C2 AD 0A 02 00 00 25 99
```

D2 – Nested Repeating Groups – MassQuote Message

D2-1 – Plain English Representation

Entry of multiple quotes for two products. First product is X with single quotes for instrument 5764 having a bid price of 96.25 and an ask price of 96.75 with a size of 10 on each side.

Second Product is Y with quotes for the two instruments 5765 and 5766. Instrument 5765 has a bid price of 96.75 and an ask price of 97.25. Instrument 5766 has a bid price of 97.00 and an ask price of 97.75.

Both quotes have a size of 10 for bid and ask. All quotes have a unique identifier.

D2-2 – FIX tag=value Encoding Representation (MsgType=”i”)

```
8=FIXT.1.1|9=264|35=i|49=TA|56=EA|34=13|52=20130222-13:34:32.128|117=6374|
296=2|
302=8453|
295=1|
299=12934|48=5764|22=8|132=96.25|133=96.75|134=10|135=10|
302=8454|
295=2|
299=12935|48=5765|22=8|132=96.75|133=97.25|134=10|135=10|
299=12936|48=5766|22=8|132=97.00|133=97.75|134=10|135=10|
10=253|
```

D2-3 – Simple Binary Encoding Message Schema

```
<sbe:messageSchema xmlns:sbe="http://www.fixprotocol.org/ns/simple/1.0"
       xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
       <types>
              <!-- Simple types -->
              <type name="counter" primitiveType="uint8" />
              <type name="uniqueId" primitiveType="uint16" />
              <type name="unsigned" primitiveType="uint32" />
              <type name="unsignedLong" primitiveType="uint64" />

              <!-- Constants -->
              <type name="FIXT11" primitiveType="char" presence="constant">FIXT.1.1</type>
              <type name="msgTypei" primitiveType="char" presence="constant">i</type>
              <type name="sender" primitiveType="char" length="2" presence="constant">TA</type>
              <type name="target" primitiveType="char" length="2" presence="constant">EA</type>
              <type name="ExchangeSymbol" primitiveType="char" presence="constant">8</type>

              <!-- Composite types -->
              <composite name="decimal32"
                      description="decimal constant exponent; only sends mantissa on the wire">
                     <type name="mantissa" primitiveType="int32" />
                     <type name="exponent" primitiveType="int8" presence="constant">-2</type>
              </composite>

              <composite name="qty8"
                     description="8 bit quantity constrained to integer values">
                     <type name="mantissa" primitiveType="int8" />
                     <type name="exponent" primitiveType="int8" presence="constant">0</type>
              </composite>
       </types>

       <sbe:message name="SimpleMassQuote" id="2" fixMsgType="i"
              description="Simplified MassQuote">
              <field name="BeginString" id="8" type="FIXT11" fixUsage="String" />
              <field name="BodyLength" id="9" type="counter" fixUsage="Length" />
              <field name="MsgType" id="35" type="msgTypei" fixUsage="String" />
              <field name="SenderCompID" id="49" type="sender" fixUsage="String" />
              <field name="TargetCompID" id="56" type="target" fixUsage="String" />
              <field name="MsgSeqNum" id="34" type="unsigned" fixUsage="SeqNum" />
              <!-- Using midnight as epoch instead of UNIX epoch, which is default -->
              <field name="SendingTime" id="52" type="unsignedLong"
                     timeUnit="millisecond" epoch="00:00" fixUsage="UTCTimestamp" />
              <field name="QuoteID" id="117" type="uniqueId" fixUsage="String" />
              <field name="NoQuoteSets" id="296" type="counter"
                     groupName="QuoteSet" fixUsage="NumInGroup" />
              <group name="QuoteSet">
                     <field name="QuoteSetID" id="302" type=uniqueId" fixUsage="String" />
                     <field name="NoQuoteEntries" id="295" type="counter"
```

```
                              groupName="QuoteEntry" fixUsage="NumInGroup" />
                       <group name="QuoteEntry">
                              <field name="QuoteEntryID" id="299" type="uniqueId"
                              fixUsage="String" />
                              <field name="SecurityID" id="48" type="uniqueId" fixUsage="String"
/>
                              <field name="SecurityIDSource" id="22" type="ExchangeSymbol"
                              fixUsage="String" />
                              <field name="BidPx" id="132" type="decimal32" fixUsage="Price" />
                              <field name="OfferPx" id="133" type="decimal32" fixUsage="Price" />
                              <field name="BidSize" id="134" type="qty8" fixUsage="Qty" />
                              <field name="OfferSize" id="135" type="qty8" fixUsage="Qty" />
                       </group>
               </group>
               <field name="CheckSum" id="10" type="counter" fixUsage="int"/>
       </sbe:message>

</sbe:messageSchema>
```

D2-4 – Simple Binary Encoded Wire Format

Annotated message

```
41                      | [H] EncodedLength = 65
02                      | [H] MessageID = 2
                        | [8] BeginString = FIXT.1.1 (constant)
                        | [9] BodyLength =
                        | [35] MsgType = i (constant)
                        | [49] SenderCompID = TA (constant)
                        | [56] TargetCompID = EA (constant)
00 00 00 0D             | [34] MsgSeqNum = 13
00 00 01 3D 02 1C C2 C0 | [52] SendingTime = 20130222-13:34:32.128
18 E6                   | [117] QuoteID = 6374
02                      | [296] NoQuoteSet = 2
21 05                   | [302] QuoteSetID = 8453
01                      | [295] NoQuoteEntries = 1
32 86                   | [299] QuoteEntryID = 12934
16 84                   | [48] SecurityID = 5764
                        | [22] SecurityIDSource = 8 (constant)
                        | [132] BidPx
00 00 25 99             |       mantissa = 9625
                        |       exponent = -2 (constant)
                        | [133] OfferPx
00 00 25 CB             |       mantissa = 9675
                        |       exponent = -2 (constant)
0A                      | [134] BidSize = 10
0A                      | [135] OfferSize = 10
21 06                   | [302] QuoteSetID = 8454
02                      | [295] NoQuoteEntries = 2
32 87                   | [299] QuoteEntryID = 12935
16 85                   | [48] SecurityID = 5765
                        | [22] SecurityIDSource = 8 (constant)
                        | [132] BidPx
00 00 25 CB             |       mantissa = 9675
                        |       exponent = -2 (constant)
                        | [133] OfferPx
00 00 25 FD             |       mantissa = 9725
                        |       exponent = -2 (constant)
```

```
0A                       | [134] BidSize = 10
0A                       | [135] OfferSize = 10
32 88                    | [299] QuoteEntryID = 12936
16 86                    | [48] SecurityID = 5766
                         | [22] SecurityIDSource = 8 (constant)
                         | [132] BidPx
00 00 25 E4              |       mantissa = 9700
                         |       exponent = -2 (constant)
                         | [133] OfferPx
00 00 26 2F              |       mantissa = 9775
                         |       exponent = -2 (constant)
0A                       | [134] BidSize = 10
0A                       | [135] OfferSize = 10
FD                       | [10] CheckSum
```

Raw message

```
41 02 00 00 00 0D 00 00 01 3D 02 1C C2 C0 18 E6 02 21 05 01 32 86 16 84 00 00
25 99 00 00 25 CB 0A 0A 21 06 02 32 87 16 85 00 00 25 CB 00 00 25 FD 0A 0A 32
88 16 86 00 00 25 E4 00 00 26 2F 0A 0A
```