Simple Binary Encoding

Release Candidate 2

Technical Proposal

March 12, 2014

v0.6

Proposal Status: Public Comment

For Global Technical Committee Governance Internal Use Only

Submission Date	Dec 18, 2013	Control Number	
Submission Status	Submitted	Ratified Date	
Primary Contact Person	Don Mendelson	Release Identifier	

r0.2

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.

DRAFT OR NOT RATIFIED PROPOSALS (REFER TO PROPOSAL STATUS AND/OR SUBMISSION STATUS ON COVER PAGE) ARE PROVIDED "AS IS" TO INTERESTED PARTIES FOR DISCUSSION ONLY. PARTIES THAT CHOOSE TO IMPLEMENT THIS DRAFT PROPOSAL DO SO AT THEIR OWN RISK. IT IS A DRAFT DOCUMENT AND MAY BE UPDATED, REPLACED, OR MADE OBSOLETE BY OTHER DOCUMENTS AT ANY TIME. THE FPL GLOBAL TECHNICAL COMMITTEE WILL NOT ALLOW EARLY IMPLEMENTATION TO CONSTRAIN ITS ABILITY TO MAKE CHANGES TO THIS SPECIFICATION PRIOR TO FINAL RELEASE. IT IS INAPPROPRIATE TO USE FPL WORKING DRAFTS AS REFERENCE MATERIAL OR TO CITE THEM AS OTHER THAN “WORKS IN PROGRESS”. THE FPL GLOBAL TECHNICAL COMMITTEE WILL ISSUE, UPON COMPLETION OF REVIEW AND RATIFICATION, AN OFFICIAL STATUS ("APPROVED") OF/FOR THE PROPOSAL AND A RELEASE NUMBER.

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

Table of Contents

Document History

Revision	Date	Author	Revision Comments
v0.1	11/18/13	Fred Malabre	Initial draft
v0.2	11/26/13	Fred Malabre	Updates based on HPWG review
v0.3	12/3/13	Fred Malabre	Move some framing content for discussion
v0.4	12/18/13	Fred Malabre	Updates based on GTC review
v0.5	2/5/14	Fred Malabre	Added the message encoding header
v0.6	3/12/14	Hanno Klein	Minor changes prior to public review

1 Introduction

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

1.1 Authors

Name	Affiliation	Contact	Role
Fred Malabre	CME Group	Fred.malabre@cmegroup.com	HPWG co-chair SBE co-lead
Don Mendelson	CME Group	Don.mendelson@cmegroup.com	SBE co-lead

2 Requirements

2.1 Business Requirements

2.1.1 Semantic

Clarify the relation between message schema, message template and message definition.
Clarify the relation between FIX tag=value data types and binary data types.

2.1.2 Versioning

One of the challenges raised with a binary encoding is the static aspect based on a given template as opposed to the dynamic aspect of the FIX tag=value encoding where tags can be added and removed on the fly.

SBE RC2 adds the concept of versioning to templates that allow decoding of messages encoded with an older version template. Specific encoding and decoding rules need to be performed to support this optional feature.

2.1.3 Byte alignment

Add the possibility to align fields on specific data structure alignment regardless of the encoding data types selected.

This would allow specific implementation to fine tune latency versus bandwidth for specific hardware.

2.1.4 Strings encoding

Clarify string encoding support for different character encodings.

Simplify the attributes for string types.

2.1.5 Decimal numbers encoding

Add support for IEEE encoding for *double* and *float* data types in addition to the mantissa and exponent data types.

2.1.6 Enumerations encoding

Enhanced RC2 to support enumerations not based on a character or a number. Some enumerations in FIX are based on strings and thus need to be mapped to an integer value in the message schema.

2.1.7 Message encoding header

Added a message encoding header for Simple Binary Encoding in complement to the session layer message framing header.

2.2 Technical Requirements

2.2.1 Semantic

The following descriptions clarify how message schema and message template are used in Simple Binary Encoding [section 1.4 of the SBE RC2 specification]:

Message schema – metadata that specifies messages and their data types and identifiers. Message schemas may be disseminated out of band. For Simple Binary Encoding, message schemas are expressed as an XML document that conforms to an XML schema that is published as part of this standard.

Message template – metadata that specifies the fields that belong to one particular message type. A message template is contained by a message schema.

In RC1, the optional XML attribute "fixUsage" was used to describe the equivalent FIX tag=value data type. The name of the attribute is changing to "semanticType" to reflect its usage within SBE templates. [section 2.1.1 of the SBE RC2 specification]
Examples:
In RC1: `<type name="short" primitiveType="int16" fixUsage="int" />`
In RC2: `<type name="short" primitiveType="int16" semanticType="int" />`

2.2.2 Versioning

Versioning is added to SBE schema to allow decoding of a binary message encoded with a newer revision of the SBE schema.

A version number is added to the XML schema [section 5.2 of the SBE RC2 specification]:

Example: `<xs:attribute name="version" type="xs:nonNegativeInteger" default="0" />`

A block length is added to each building blocks of a message (root level and repeating groups) [section 5.2.3 of the SBE RC2 specification]:

Example: `<message name="FIX Binary Message1" id="1" blockLength="4">`

The following constraints are added is order to support versioning [section 5.4 of the SBE RC2 specification]:

- The attribute "blockLength" must be used to skip to the next set of data and skip unknown data added on a newer revision of a template.
- The template must be updated accordingly to only add new fields at the end of a block on a newer revision.

Examples:

Initial version of a message schema

```
<messageSchema package="FIXBinaryTest"
   byteOrder="littleEndian">
    <types>
         <type name="int8" primitiveType="int8"/>
    </types>
    <message name="FIX Binary Message1" id="1" blockLength="4">
         <field name="Field1" id="1" type="int8" semanticType="int"/>
    </message>
</messageSchema>
```

Second version – a new message is added

```
<messageSchema package="FIXBinaryTest" byteOrder="littleEndian"
   version="1">
    <types>
         <type name="int8" primitiveType="int8"/>
         <type name="int16" primitiveType="int16"
                     sinceVersion="1"/>
    </types>
    <message name="FIX Binary Message1" id="1" blockLength="4">
         <field name="Field1" id="1" type="int8" semanticType="int"/>
    </message>
    <!-- New message added in this version-->
    <message name="FIX Binary Message2" id="2" blockLength="4"
sinceVersion="1">
         <field name="Field2" id="2" type="int16" semanticType="int"/>
    </message>
</messageSchema>
```

Third version – a field is added

```
<messageSchema package="FIXBinaryTest" byteOrder="littleEndian"
   version="2">
    <types>
         <type name="int8" primitiveType="int8"/>
         <type name="int16" primitiveType="int16"
               sinceVersion="1"/>
         <type name="int32" primitiveType="int32"
         sinceVersion="2"/>
    </types>
    <message name="FIX Binary Message1" id="1" blockLength="8">
         <field name="Field1" id="1" type="int8" semanticType="int"/>
         <field name="Field11" id="11" type="int32" semanticType="int"
         sinceVersion="2"/>
    </message>
    <message name="FIX Binary Message2" id="2" blockLength="4"
         sinceVersion="1">
         <field name="Field2" id="2" type="int16" semanticType="int"/>
    </message>
</messageSchema>
```

2.2.3 Byte alignment

An offset can be added to all fields in order to provide a direct access at specific locations independent from the size of a given field. [section 3.3.2 of the SBE RC2 specification]

Example: `<field name="Side" id="54" type="char" offset="14" semanticType="char"/>`

2.2.4 Strings encoding

Define strings as an array of octets as opposed to an array of char. [section 2.7 of the SBE RC2 specification]

The name of the attribute is changed to “characterEncoding” in the XML schema to specify the encoding used. The default encoding is UTF-8.

Use the attribute “length” with a value set to “0” (zero) to indicate variable length strings instead of the Boolean attribute “variableLength”.

Example: `<type name="varData" length="0" primitiveType="uint8" semanticType="data" characterEncoding="UTF-16"/>`

2.2.5 Decimal numbers encoding

Added support for IEEE Standard for Floating-Point Arithmetic (IEEE 754) encoding for decimal numbers. [section 2.6 of the SBE RC2 specification]
Examples:

```
<type name="float" primitiveType="float" />
<type name="double" primitiveType="double" />
```

2.2.6 Enumerations encoding

Clarify ways to describe enumerations in the XML schema, either by reference or direct.

Clarify cross reference for constant values in enumerations.

Clarify lookup structure for string based enumerations.

[section 2.12 of the SBE RC2 specification]

Example:

```
<enum name="PartyIDSourceEnum" primitiveType="char">
      <validValue name="BIC">B</validValue>
      <validValue name="GeneralIdentifier">C</validValue>
      <validValue name="Proprietary">D</validValue>
</enum>
<field type="PartyIDSourceEnum" name="PartyIDSource" id="447"
description=”Party ID source is fixed” presence="constant"
valueRef=”GeneralIdentifier” />
```

2.2.7 Message encoding header

Four fields are added as the message encoding header for Simple Binary Encoding messages [section 3.2 of the SBE RC2 specification]:

- **Block length of the message root**—the total space reserved for the root level of the message not counting any repeating groups or variable-length fields.
- **Template ID**—identifier of the message template
- **Schema ID**—identifier of the message schema that contains the template
- **Schema version**—the version of the message schema in which the message is defined.

3 Issues and Discussion Points

3.1.1 Schema validation

Using XML as a message template and the message structure being complex, standard XML validation tools can not be used to validate a schema fully compliant with SBE.

Additional tools will need to be created to fully validate an XML schema in compliance with SBE.

3.1.2 String with no value

Currently, there is no distinction between a string with no value and an empty string (i.e. with a length of zero).

If there is really a need to make this distinction, a subsequent release of SBE will need to handle this.

4 References

Reference	Version	Relevance	Normative
FIX Simple Binary Encoding RC1 Specifications	Final	Full specification as approved for RC1 in June 2013 by the FPL GTC.	
Simple Binary Encoding – Release Candidate 2	RC2 v0.21	Full specifications based on RC1 with the addition of the technical solutions from this document.	
SimpleBinary-ReleaseCandidate2	Draft	Full XSD supporting the specifications .	

5 Relevant and Related Standards

Related Standard	Version	Reference location	Relationship	Normative
None				

6 Intellectual Property Disclosure

Related Intellection Property	Type of IP (copyright, patent)	IP Owner	Relationship to proposed standard
None			

7 Definitions

Term	Definition

8 Simple Binary Encoding

8.1 Specifications

Full specifications for the Simple Binary Encoding are available in separate document (*FIX Simple Binary Encoding – Release Candidate 2, revision RC2 0.21*). The standard defines wire format and message schema declaration.

8.2 Schema

An XML schema (XSD) is provided to standardize XML message schemas. The XSD file should published to users with the specification document. For this release, the name of the XSD file is SimpleBinary-RC2.xsd.

Appendix A - Usage Examples

Examples are provided in the specification document.

Appendix B – Compliance Strategy

Message schemas should be validated against the provided XML schema (XSD).