FIX PROTOCOL
INDUSTRY-DRIVEN MESSAGING STANDARD℠

Encoding FIX using Google Protocol Buffers

Release Candidate 2 – User Guide

RELEASE CANDIDATE 2 Document Revision 0.22: March 27, 2014

THIS DOCUMENT IS A RELEASE CANDIDATE FOR A PROPOSED FIX TECHNICAL STANDARD. A RELEASE CANDIDATE HAS BEEN APPROVED BY THE GLOBAL TECHNICAL COMMITTEE AS AN INITIAL STEP IN CREATING A NEW FIX TECHNICAL STANDARD. POTENTIAL ADOPTERS ARE STRONGLY ENCOURAGED TO BEGIN WORKING WITH THE RELEASE CANDIDATE AND TO PROVIDE FEEDBACK TO THE GLOBAL TECHNICAL COMMITTEE AND THE WORKING GROUP THAT SUBMITTED THE PROPOSAL. THE FEEDBACK TO THE RELEASE CANDIDATE WILL DETERMINE IF ANOTHER REVISION AND RELEASE CANDIDATE IS NECESSARY OR IF THE RELEASE CANDIDATE CAN BE PROMOTED TO BECOME A FIX TECHNICAL STANDARD DRAFT.

March 2014

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.

Document History

Revision	Date	Author	Revision Comments
0.1	2013-04-10	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Alessandro Triglia, OSS Nokalva	Initial draft.
0.2	2013-04-16	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Alessandro Triglia, OSS Nokalva	Changes due to feedback from HPWG review.
0.3	2013-08-15	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Joe Wood, Credit Suisse	Updated with conclusions from Community Review.
0.4	2013-11-05	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Joe Wood, Credit Suisse	Release Candidate 2 initial draft
0.15	2013-11-22	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Joe Wood, Credit Suisse	Submitted to GTC.
0.16	2014-1-27	Greg Malatestinic, Jordan & Jordan	Changes based on feedback from GTC.
0.22	2014-3-27	Greg Malatestinic, Jordan & Jordan Sara Rosen, EBS Joe Wood, Credit Suisse	Changes based on introduction of simple open framing header and message encoding headers.

Table of Contents

1 Introduction

The High Performance working group (HPWG) was created by the FIX Protocol Limited (FPL) Global Steering Committee in July 2012. Fred Malabre and Mark Reece are co-chairing the working group. The charter of the working group is to identify opportunities to enrich high performance financial messaging and propose specific enhancements to FIX to address these identified opportunities, including application level, session level (recovery), and encoding.

Several encoding subcommittees were formed. Among them is The Google Protocol Buffers encoding subgroup, led by Sara Rosen and Greg Malatestinic.

The mandate of the Google Protocol Buffer encoding subgroup was to define a mechanism for mapping FIX to Protocol Buffers which can support the rich semantics of the FIX language while meeting the performance goals of the High Performance trading community.

The Google Protocol Buffer encoding utilizes existing industry standard encodings and applies them to the FIX domain. Google Protocol Buffers is a mature binary encoding which can be applied “out of the box” to support FIX language semantics. However, the richness of this technology supports multiple solutions to mapping the existent FIX data types to a binary protocol. In the interest of standardization, this document will specify a normative encoding of the FIX data types in protocol buffers for interoperable exchange of financial data over FIX. Additionally, this document will spell out best practice guidelines for maximizing the efficiency of protocol buffer encodings.

1.1 Google Protocol Buffers for FIX

Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data.

Protocol buffers are an attractive encoding technology for FIX messages for a number of reasons:

- Open – The complete GPB specification has been published and Google has clearly stated that GPB is available for use by anyone. Protobuf is language and platform independent, supporting C++, Java, Python and other languages on most common operating systems. Clients do not need to write their own encoders/decoders, but can rather incorporate the Google provided API’s within their software.
- Flexible - Like FIX, protobuf messages are tagged. This supports optional fields, which can be leveraged to support multiple use cases via a single message definition.
- Versioning - Protobuf messages support backward compatibility across older versions of message specifications. Messages can be modified without breaking applications that utilize prior message definitions, thus reducing deployment dependencies between message producers and consumers.
- Performance - Protobuf produces compact binary messages to maximize network bandwidth and reduce serialization delays. Protobuf serialization is CPU efficient, with a low encoding and decoding overhead and a minimal footprint.
- Ease of Use - Protobuf is well documented and easy to learn. Encoders and decoders can be embedded directly in the target applications, for “do it yourself” FIX implementations.

- Structured - Protobuf supports repeating fields and nested messages, similar to FIX repeating groups. Unlike FIX, nested messages can have the same field names at any level of nesting, so there is no need for artificial constructs like Parties, NestedParties, NestedParties2, etc.
- Templates - Similar to FAST[SM] (http://www.fixtradingcommunity.org/pg/structure/tech-specs/fast-protocol), protobuf message structures are defined in message templates, which are used by message recipients to decode the data. Default values can be populated based on template definitions, further reducing messaging bandwidth for the most commonly used values.
- Adoption - Protobuf has a large and active user community. It is also a mature technology, in use since 2001. It also serves as the underlying messaging technology for a number of open source frameworks.

1.2 References

[1] https://developers.google.com/protocol-buffers/docs/overview
GPB project home – Language Guide and Encoding Specification.

[2] Specification for mapping the FIX Unified Repository to GPB. (work in progress)

[3] http://www.fixtradingcommunity.org/pg/structure/tech-specs/fast-protocol
FIX Adapted for Streaming[SM] (FAST[SM]) specifications.

[4] FIX Simple Open Framing Header Specification (work in progress)

2 Google Protocol Buffers Overview

2.1 Templates

GPB data structures are defined as "messages" in a ".proto" template file. The template provides a machine-readable interface definition which is input to language-specific code generators to produce message encoders and decoders. These expose getters and setters for each field as well as methods to serialize/parse the whole structure to/from raw bytes. Senders and receivers must each have a copy of the proto file to encode or parse the message. These are typically exchanged out-of-band. Dynamic transmission is also possible, though less performant.

The template defines the structure and properties of the message and its fields. Messages may be nested or contain repeating groups of messages. Fields may be specified as required or optional, and may contain default values. Templates may also contain human-readable comments and namespace identifiers to prevent message collisions between packages. A single proto file may define multiple message types, and can also reference messages defined in external proto files, via the "import" command.

Here is a sample template for a FIX Logout message:

```
message Logout { // FIX MSG_ID = 5
   optional Header header = 1;
   optional int32 session_status = 2 [default = 4];  //Session status at time of logout
   optional string text = 3;
   optional bytes encoded_text = 4;
   optional Trailer trailer = 5;
};
```

2.2 Field Properties - Names and Tags

The general structure of a protobuf field is

<rule> <type> <name> = <tag> {[<options>]};

Every protobuf field has an ASCII field name and an integer field tag. The field name is the data structure's human-readable symbolic name. Field names are not present in the binary payload of the message. Rather, the field name is mapped to the field's accessor methods to encode/decode the binary message from/to the data structures specified in the message's proto template. For FIX encoding, the protobuf field names are based on the FIX field and component names.

The field tag is the machine-readable integer which is embedded in the binary message, and which enables the GPB decoder to map the value from the wire-stream value to its associated data structure.

Tags with a value of 1-15 occupy one byte. Tags in the range 16-2047 take two bytes, and larger values take more. A highly optimized schema will determine the most frequently occurring message elements and assign these to the lower-value tags.

For this reason, and also to support mapping a single FIX field to multiple protobuf fields, the GPB Encoding Subgroup has decided not to utilize the FIX tags for the protobuf tag ids. Rather, tag numbers were sequentially auto-generated based on field order in the FIX repository. Clients who wish to optimize bandwidth utilization may re-sequence the fields based on the relative field utilization in their context. Unlike FIX tag numbers, protobuf field tags need not be globally unique, but must only be unique per message. The standard convention is to utilize the lowest tag numbers as possible so as to minimize the size of an encoded message.

Field rules are "required", "optional", and "repeated". Required fields must appear exactly once. Optional fields may appear zero or once, and repeated fields may appear zero to any number of times.

2.3 Message Encoding

A top level message has no envelope structure. The message is just a sequence of its member key-value fields. Embedded messages are encoded similar to strings, e.g. they contain a length-delimited sequence of bytes. Like all fields, this embedded message is preceded by its field tag, which associates it with its message type. This allows the protobuf decoder to recursively invoke the appropriate parser for the byte string containing the embedded message.

Example of a simplified Market Data Snapshot message:

```
Message Instrument {
        string symbol = 1;
        fixed32 security_id = 2;
}

Message MDFullGrp {

…

}

Message MarketDataSnapshotFullRefresh {
        fixed32 md_request_id = 1;
        Instrument  instrument =2;
        MDFullGrp md_full_grp =3;
}
```

When parsing the MarketDataSnapshotFullRefresh message, presence of tag=2 causes the parser to extract the length-delimited sequence of bytes representing the instrument field. It then invokes the parser of the Instrument class to decode the nested message.

2.4 Optional Fields

Protobuf is a tagged-value encoding. This means that the presence or absence of a field can be detected by the presence of the field tag in the output stream. As a result, Protobuf is especially efficient for encoding /decoding messages with sparse optional fields, i.e. messages where a large number of fields are empty. Optional fields which are not set by the encoder carry no bandwidth overhead.

There are no “null” values in protobuf. When a message is parsed, if an optional field is not present in the message, the GPB decoder sets the value of the associated field to its default value. Default values may be specified explicitly in the proto template. Otherwise, a default value is determined implicitly, based on field type.

It is occasionally useful to determine whether the value of a decoded field was explicitly set by the message producer or whether it was set to a default value by the decoder. For each optional field, XXX, the GPB pre-compiler generates a Boolean accessor method, HasXXX(), which can be utilized to determine whether or not the field was present on the wire.

2.5 Repeating groups

The GPB encoding supports repeating groups. Repeating groups may appear any number of times in the message, including zero times. The order of the repeated group will be preserved in the protocol buffer. A repeating group with a recurrence of zero does not consume message bandwidth.

GPB also supports a packed encoding of repeating groups. Only repeating groups which contain only primitive numeric types can be declared packed. Rather than having the tag and type appear for each element of the repeating group, the entire list is encoded as a single length-delimited key-value pair containing the concatenated values of all the data elements.

Source: https://developers.google.com/protocol-buffers/docs/encoding#optional

2.6 Protobuf Options

GPB encoding makes use of additional metadata in the form of GPB Options. Options are annotations used to add information about the message schema over and above the standard schema constructs offered by the proto schema file. This metadata can be used to decorate fields, messages, packages, enumerated types and enumerated values. Conceptually, GPB Options are similar to Java’s decorators or .NET’s attributes.

Protocol Buffers provide a set of predefined Options that can be used in any proto file. As illustrated below, the predefined set includes a file option to specify the generated Java package name, and field options to specify a default value or to indicate that a field is deprecated. These options are added in-line to the proto file, with field options appended after the field declaration in square parenthesis:

```
package sample;
option java_package = "com.fix.sample";
message SampleMessage
{     …
      optional int32 old_field = 6              [deprecated=true];
      optional int32 new_field=7                [default=1000];
}
```


2.7 Custom Options

In addition to the predefined set of options supplied by Google, custom options can be defined to expose additional metadata via the schema. Metadata can be used to annotate the .proto file with user interface bindings, expected field ranges, or custom validations. For FIX encoding via GPB, metadata is used to annotate the proto schema with the FIX tag numbers, FIX version added/deprecated, and FIX enumerated values from the FIX Repository. Rather than embedding this information in comments or rules of engagement documents, custom options provide programmatic access to this rich metadata.

Custom options are typically defined in their own proto file as extensions to the core GPB metadata types. For example, the following file defines a file and a field custom option. Note that the Google-supplied file **descriptor.proto** needs to be imported into this file as it supplies the definitions of the core GPB metadata types:

```
import "google/protobuf/descriptor.proto";
package my_options;
extend google.protobuf.FieldOptions {
    optional sfixed32 max_value = 51234;
}
extend google.protobuf.MessageOptions {
    optional string message_type =51236;
}
```

These custom options can then be used in proto files by importing their definitions and using them to annotate files, messages and fields. The custom option's name is required to be placed inside parentheses to differentiate it from the predefined standard options:

```
import "my_options.proto";
package sample;
option java_package = "com.fix.sample";
message TradeCaptureReport
{
    option (my_options.message_type) = "AE";
    optional string trade_id = 1;
    optional fixed64 last_qty =2[(my_options.max_value) = 999999999];
}
```

Metadata defined by GPB Options and Custom Options can be accessed using the protobuf library **Descriptor** class. Options provide an efficient way to supply encoding information with no overhead to the message payload.

The FIX to GPB mapping process makes use of GPB Options to provide:

- Information tying proto file entries back to the FIX repository.
- FIX version information, tying fields and enums to the FIX Repository version when they were introduced or deprecated.

- Decimal precision of fixed and floating point numbers
- TimeUnit and epoch of dates and timestamps
- Validation information, such as string length and valid ranges

Just like GPB messages, custom option definitions may be organized into multiple .proto files, each with its own namespace. The FIX to GPB mapping specification defines two custom option .proto files:

- meta.proto – general metadata describing field usage
- fix.proto - FIX-specific metadata, based on the information in the FIX Repository

For more information about GPB Custom Options refer to https://developers.google.com/protocol-buffers/docs/proto#options, or the complete Protocol Buffers Documentation [1].

2.8 Embedded Comments

Comments embedded within a .proto schema are propagated into the generated source code. This is a convenient method to supply message documentation to the IDE of the protobuf-generated classes.

For FIX encoding, comments from the FIX repository as well as custom usage comments can be embedded within the proto files to make this documentation available to message consumers.

Example:

```
optional bool working_indicator = 21    [(fix.tag)=636, (fix.type)=BOOLEAN,
                                        (fix.added)=FIX_4_3], (doc.version="2.7.4");
                                        // Indicates if the order is currently being worked.
                                        Applicable only for OrdStatus = "New". For electronic
                                        markets, indicates that the order has transitioned from
                                        a contingent order to a market order.
```

2.9 Utilities

Google provides GPB parsers for Java, C++, and Python. Also provided are utilities for printing and displaying GPB messages in a human-readable, text-based format. See for example the C++ text formatting methods at: https://developers.google.com/protocol-buffers/docs/reference/cpp/google.protobuf.text_format

Many other language encoders/decoders have been developed by the open-source community. These include ActionScript, C#, Javascript, Lua, Perl, PHP, R, Android, and Ruby.

Other open-source GPB utilities include tools for converting GPB messages to XML, JSON, or HTML.

A full list of Third Party GPB Add-Ons is provided at: http://code.google.com/p/protobuf/wiki/ThirdPartyAddOns

3 FIX Mapping

The following is a brief discussion of a few of the salient points of the mapping. A full description of the procedure to map the full contents of the FIX Unified Repository to GPB can be found at hyperlink.

3.1 Naming conventions

Field names in the proto file are recommended to conform to the GPB field naming conventions. This will allow the GPB language-specific compilers to generate variable names which conform to the naming conventions of each target language.

GPB field names are lower case underscore-separated.

FIX field name	proto field name	generated Java variable	generated .NET variable	C++
OnBehalfOfCompID	on_behalf_of_comp_id	onBehalfOfCompId	OnBehalfOfCompId	on_behalf_of_comp_id
IOINaturalFlag	ioi_natural_flag	ioiNaturalFlag	IoiNaturalFlag	ioi_natural_flag
MDEntryPx	md_entry_px	mdEntryPx	MdEntryPx	md_entry_px

GPB message names are in Pascal Case, which capitalizes the first character of each composite word, including acronyms. This applies to both stand-alone FIX messages and FIX components embedded within GPB messages.

FIX message or component name	proto message name	generated Java class	generated .NET class	C++
NewOrderSingle	NewOrderSingle	NewOrderSingle	NewOrderSingle	NewOrderSingle
MDIncGrp	MdIncGrp	MdIncGrp	MdIncGrp	MdIncGrp
IOIQualGrp	IoiQualGrp	IoiQualGrp	IoiQualGrp	IoiQualGrp

Instances of the embedded messages are themselves GPB fields, and follow the lower-case underscore-separated convention.

Example: The FIX component “PtysSubGrp” is mapped to a GPB message type of the same name.

```
message PtysSubGrp {
        optional string party_sub_id = 1;
        optional PartySubIdTypeUnion party_sub_id_type = 2;
}
```

Instances of this message type are embedded in other GPB messages as “ptys_sub_grp”.

```
message Parties {
        optional string party_id = 1;
```



```
        optional PartyIdSourceEnum party_id_source = 2;
        optional PartyRoleEnum party_role = 3;
        repeated PtysSubGrp ptys_sub_grp = 4;
    }
```

FIX enumeration types are mapped to GPB enums. These follow the same convention as other GPB message types (PascalCase), concatenated with the string "Enum", e.g. PartyRoleEnum.

Enumerated-value names are fully upper case, underscore-separated names. The name of the GPB enumerated type (minus the "Enum") is pre-pended to the name of the GPB enumerated value, separated by yet another underscore.

```
enum HandlInstEnum {
    HANDL_INST_AUTOMATED_EXECUTION_NO_INTERVENTION = 0;
    HANDL_INST_AUTOMATED_EXECUTION_INTERVENTION_OK = 1;
    HANDL_INST_MANUAL_ORDER = 2;
}
```

Enumerated values are prepended with the name of the enumeration due to the GPB restriction that all enumerated values in a single proto be unique. For example, both the ExecType and OrdStatus enumerations have an enumerated value of "CANCELED". Following the convention above, the two enumerated values are EXEC_TYPE_CANCELED, and ORD_STATUS_CANCELED, which preserves their uniqueness.

3.2 Automated Mapping from FIX Repository

The FIX to GPB Mapping Specification [2] describes how to take as input an `XML` element, whose syntax and semantics comply with those of the `<fix>` element of the FIX Unified Repository, and generate a Google Protocol Buffers schema, consisting of a set of GPB message and enumeration types.

This specification provides rules for mapping FIX messages and component blocks to GPB messages, FIX fields to GPB field definitions, and enumerated field values to GPB enum definitions. It shows that for each FIX message or component, the mapping of its field items will depend on their effective types which include but are not limited to the following:

- Component block
- Repeating block
- Datatype restricted by the enumeration specified in `<enum>` child elements
- Datatype with no restricted values
- Multiple-value type with values restricted by the enumeration specified in `<enum>` child elements
- Multiple-value type with no restricted values

The mapping utilizes various encoding attributes to annotate repository elements such as <datatype>, <field> and <fieldRef>. These attributes are used to determine the most favorable encoding when there are several options. They include:

- minValue - minimum permitted value of the integer datatype
- maxValue - maximum permitted value of the integer datatype

- minLength - minimum permitted length of the string datatype
- maxLength - maximum permitted length of the string datatype
- numBits - maximum size in bits of the data element
- isNumeric– indicates that a FIX string will take only numeric values
- isFloat – indicates that a numeric value should be encoded as a binary float rather than as an integral mantissa and exponent
- isFixedPoint – indicates that a consistent exponent should be applied to the integral mantissa
- exponent - the fixed exponent of a fixed-point decimal number or the default exponent for a floating-point decimal number

To illustrate how the effective datatype influences the mapping, let's consider the FIX field "Side". Since it is a char type one may think it should be mapped to the GPB bytes type which is used to encode char values. But since "Side" is restricted by enumerated values "Buy", "Sell", "Sell Short", etc., it's effective datatype causes the following GPB enum type to be generated:

```
enum SideEnum {
        SIDE_BUY = 0;
        SIDE_SELL = 1;
        SIDE_BUY_MINUS = 2;
        SIDE_SELL_PLUS = 3;
        SIDE_SELL_SHORT = 4;
        SIDE_SELL_SHORT_EXEMPT = 5;
        // etc.
}
```

Any FIX message or component containing the Side field will generate a GPB message containing a field definition of field type "SideEnum". For example, the FIX message NewOrderSingle will map to the following proto snippet:

```
message NewOrderSingle {
        // …
        optional SideEnum side = n;
        // …
}
```

To illustrate how encoding attributes may influence the mapping, let's consider the FIX integer field, LiquidityNumSecurities, with the encoding attributes *minValue* equal to 0 and *maxValue* equal to 1000. We can conclude that all values of this field are positive and only two bytes are needed to store its value. This is mapped to the GPB type fixed32. Had *maxValue* been a number larger than 2^32 this would map to the GPB type fixed64.

Please refer to the specification, [2], for complete details of the mapping.

3.3 FIX Metadata

Attributes derived from the FIX repository and encoding attributes may be used to embellish the proto schema with metadata in the form of custom options. This metadata helps define message usage and tie the GPB proto files back to the FIX repository.

As an example, the following field and data-type definitions from the FIX repository are shown below. A fragment of the Repository's **fields.xml** shows the definition of two FIX fields, with some information removed for simplicity:

```
<field id="120" name="SettlCurrency" type="char" textId="FIELD_120" added="FIX.4.0"
          notReqXML="1">
    <EncodingInfo isFixedPoint="true" exponent="-6"/>
</field>
<fieldRef id="99" name="StopPx" required="0" legacyPosition="35" legacyIndent="0"
              added="FIX.2.7"
              textId="MSG_9_REF_99">
    <EncodingInfo isFixedPoint="true" exponent="-6"/>
</ fieldRef>
```

The above two fields appear in the ListOrdGrp component block, which maps into a GPB message fragment as follows:

program-trading.proto:

```
    …
message ListOrdGrp {
    …
    optional sfixed64 stop_px = 27  [(fix.tag)=99, (fix.type)=PRICE, (fix.field_added)=FIX_2_7];
    optional sfixed32 stop_px_exponent = 28 [default=-6];
    optional string settl_currency = 29    [(fix.tag)=120, (fix.type)=CURRENCY, (meta.minLength)=3,
                                (meta.maxLength)=3,(fix.field_added)=FIX_4];
    …
}
```

As can be seen from the above example:

- The FIX tag is indicated using the **fix.tag** field option.
- The FIX datatype is indicated using the **fix.type** field option.
- The fixed point price field is split into two GPB fields, one representing a mantissa, and the second representing its exponent.
- The Currency string has a constant length of 3. This is exposed through the **minLength** and **maxLength** field options.

The mapping process uses the FIX repository and optional encoding attributes to generate GPB schemas decorated with these custom options. The complete set of encoding attributes and corresponding custom options can be found in the GPB Mapping Specification [2].

The custom options used by the FIX GPB encoding are split into two file separate files, meta.proto and fix.proto:

- **meta.proto** metadata is related to general binary message encoding (e.g. minSize, timeUnit)
- **fix.proto** metadata is specific to the FIX repository (e.g. FIX msgType, FIX tag, FIX version added)

The table below describes each custom option and its usage:

Option	Applied To	Defined In	Description
time_unit	A date or date/time field	meta.proto	Defines the unit of time of the applied scalar field. e.g. TIME_UNIT_DAYS, TIME_UNIT_MILLISECONDS.
epoch	A date or date/time field	meta.proto	The starting reference point used to calculate the date or date/time offset.
exponent	A fixed or floating point scalar field	meta.proto	Defines the exponent value (base 10) of the fixed or floating point field.
minValue, maxValue	GPB scalar field	meta.proto	The minimum and maximum expected value of the field.
minLength, maxLength	GPB string field	meta.proto	The minimum and maximum length of a string. When minLength= maxLength, indicates that the string is fixed length.
msg_type	Top-level GPB message	fix.meta	Indicates the FIX message type the GPB message has been generated from.
tag	Any GPB field	fix.meta	Indicates the FIX tag the field was generated from.
type	Any GPB field	fix.meta	Indicates the FIX Repository data type of the field
field_added	Any GPB field	fix.meta	Indicates the version of FIX in which the field was first added to the component or message
field_added_ep	Any GPB field	fix.meta	Indicates the extension pack (if any) in which the field was first added to the component or message
field_deprecated	Any GPB field	fix.meta	If a field has been deprecated for a given component or message, this field option indicates

			in which version of FIX it was first deprecated .
enum	GPB enumerated value	fix.meta	Indicates the corresponding enumerated value in the FIX repository.
enum_added	GPB enumerated value	fix.meta	Indicates the version of FIX when the enumerated value was added to the component or message.
enum_added_ep	GPB enumerated value	fix.meta	Indicates the extension pack (if any) of FIX when the enumerated value was added to the component or message.

3.4 Proto File Organization

Rather than mapping the entire FIX repository to a single unwieldy proto file, recommendation is to partition the set of FIX messages into namespaces paralleling the 26 FIX categories. This allows each FIX category to be implemented and versioned independently.

In the auto-mapped proto files generated from the FIX repository, the set of FIX GPB schema files are comprised of a set of static files for custom options and supporting data-types, and the files generated from the FIX repository. The generated files are organized into files named after the containing FIX category. All fields, messages, and data-types belonging to a single FIX category are contained in a single proto file. The name of the proto file is taken from the category name, but converted to lower-case with hyphen separators (e.g. single-general-order-handling.proto). In addition to the category name, proto files should be identified with their version numbers .e.g. single-general-order-handling.v3.proto. Version numbers will be used in the GPB fixed-length message prefix, and should therefore be integer numbers.

Dependencies between .proto files are managed by declared imports. When a GPB field declared in one .proto file is referenced from within another .proto file, the name of the field must be qualified by its originating namespace.

The core set of static GPB schema files are as follows:

- **meta.proto** – general custom options used to annotate the schema
- **fix.proto** – FIX specific custom options derived from the FIX repository
- **google/protobuf/descriptor.proto** – required Google metadata for field options (available from protobuf download site)

The following two proto files are imported as a dependency into each generated GPB schema file:

- **common.proto** – contains common data types and components used by multiple categories
- **session.proto** – contains the definition of the FIX standard header and trailer

For a full list of FIX categories and their associated proto file names, see Appendix A.

3.5 Field Ordering

For backwards compatibility, it is important that as new fields are added to the FIX Repository, the newly generated GPB field numbers do not collide with the existing field numbers of previous generations of proto files. Generated GPB schema files must therefore ensure that newer fields are appended at the end of GPB data-structures so that the existing sequence of field numbers is not disrupted.

The ordering logic utilized by the FIX Repository Automated Mapping Specification [2] makes use of the FIX repository's "added " and "ep_added" attributes to sequence fields based on the FIX version when they were added to the repository. Fields and enumerated values which were added within the same FIX repository revision (indicated by the version, service pack and extension pack) are added in alphabetical order.

Just as it is important that new fields are added to the repository with an explicit FIX version, it is also important that defunct fields be marked as deprecated, rather than deleted. Fields should never be removed from the repository, as this would disrupt the field ordering and break backwards compatibility.

Organizations which generate proto files from Custom Repositories should implement a similar methodology to ensure that new fields are added at the end of previously defined GPB messages. Rather than using the FIX version when the field was introduced, ordering can be based on a custom encoding attribute indicating the proprietary version when the field was added.

4 FIX Data Types

4.1 Timestamps

ASCII timestamps are particularly inefficient. For example, a millisecond timestamp in the format YYYYMMDD-HH:MM:SS.sss consumes 21 bytes, and must be converted to a numeric values for date comparisons or calculations.

In mapping FIX to GPB, timestamps are represented by an integer offset, based on an epoch and a timeunit. Timeunits may be days, hours, minutes, seconds, milliseconds, microseconds, nanoseconds, or picoseconds. The epoch is typically Jan 01, 1970, although other epochs may be defined for special purposes. The size of the integer (32 or 64 bits) will depend on the choice of timeunit and epoch.

Timeunit and epoch are not sent over the wire, but are rather specified as metadata in the proto file and in an organization's Rules of Engagement document. The FIX repository encoding attributes "epoch" and "timeUnit" may be used to automatically generate an appropriately sized integer for the timestamp offset.

4.2 Decimal Prices

A special challenge exists for mapping FIX decimal prices to binary data. Although all binary encodings support floating point binary numbers, the general convention in the financial industry is avoid floating point representation of decimal prices due to rounding issues. This is avoided in the existing FIX tag/value encoding by recording decimal prices as ASCII strings. However, this carries a heavy performance penalty, both in terms of bandwidth utilization and processing overhead to compare or sort ASCII prices.

The solution implemented for FIX over GPB is to represent decimal prices by a pair of integers – a mantissa and an exponent, where the exponent determines placement of the decimal point.

$$Decimal\ value = Mantissa \times 10^{exponent}$$

For example, to represent the decimal price of 12.3456, mantissa=123456 & exponent=-4.

When the constant value of the exponent is known at the time of proto definition, the exponent can be included as a custom option on the proto field where it will be accessible to decoders. This approach should be taken only when it is foreseen that there will be no variability in exponent value.

```
optional sfixed32 last_px = 1 [meta.exponent=-2] ;  // two decimal digits
```

A safer approach is to set the expected value of the exponent as the default value of a corresponding exponent field

```
optional sfixed32 last_px = 1 ;
optional sfixed16 last_px_exponent = 2 [default=-6];
```

When the field's exponent value is the same as its default value, it need not be sent on the wire. But, for exception cases, the default can be overridden as needed. Note that by GPB convention, even when no default is explicitly set, all integers have an implicit default value of zero. In the case of an exponent field, this would indicate an integral number without any decimal places.

Organizations are encouraged to carefully consider current and future usage scenarios before deciding on a default exponent. Mid-prices, reference rates, and other derived market data fields will often require extra digits of precision. It is better to err on the side of allocating extra digits of precision. Recall that a change to a default value is a breaking change which would render the proto definition not backward compatible, since encoders and decoders must access the same default value.

Occasionally, fractional prices may best suit the domain model, for example when quoting in thirds, e.g. $123\ ^{2}/_{3}$. Then the price should be declared as a binary type float:

```
optional float last_px =1;
```

The FIX Repository may be annotated with encoding attributes to drive the automated mapping of FIXPrice fields to the desired GPB encoding. The relevant encoding attributes are: “isBinaryFloat” “isFixedPoint “, “exponent”, “numBits”, “minValue”, and “maxValue”. See section 3.5 below for more information on the usage of encoding attributes to auto-generate GPB proto files from a FIX Repository.

4.3 MultipleCharValue

FIX fields of type MutilpleCharValue (or MultipleStringValue) are mapped to a sequence of enumerated values, using a GPB “repeated group.” Inclusion of each element of the repeating group indicates selection of its associated optional value.

For example, the FIX field, ExecInst, is a MultipleCharValue field where the choice of values is selected from a pool of enumerated values. This maps to the GPB field definition:

```
repeated ExecInstEnum exec_inst = 21 [packed = true];
```

where the enumerated values are defined in the following GPB enum:

```
enum ExecInstEnum {
        EXEC_INST_STAY_ON_OFFER_SIDE = 0;
        EXEC_INST _NOT_HELD = 1;
        EXEC_INST _WORK = 2;
        EXEC_INST _GO_ALONG = 3;
        // ...
}
```

Note that repeating groups of enumerations can take advantage of the optimized “packed” encoding option for repeating groups, since they contain no complex data structures.

5 Usage Guidelines

5.1 Optional/Required Fields

Google recommends that all fields be marked as optional. This is especially important to support message evolution.

> **"Required Is Forever** -You should be very careful about marking fields as required. If at some point you wish to stop writing or sending a required field, it will be problematic to change the field to an optional field – old readers will consider messages without this field to be incomplete and may reject or drop them unintentionally. You should consider writing application-specific custom validation routines for your buffers instead. Some engineers at Google have come to the conclusion that using required does more harm than good; they prefer to use only optional and repeated. However, this view is not universal." https://developers.google.com/protocol-buffers/docs/proto

Consequently, the GPB Encoding subgroup recommends that all FIX fields in the protobuf message definition be marked as optional. Semantically, fields can still be required, but enforcement of required message fields should be done on the application level, rather than by the GPB parser. Rules of engagement documents should be used to specify which fields are required in a given context. When such a "required" field is missing, the message should be rejected with an application or session level rejection message.

One exception to the above guideline is when a set of fixed-length fields at the beginning of a message are designated as "Direct Access" fields, often used for high-performance routing. Since the Direct Access mechanism depends on all such fields being present in every message, these fields may be tagged as "required" in order to guarantee their presence in every message.

5.2 Versioning

Protobuf supports both forward and backward compatibility of message specifications, so that message senders and receivers do not have to utilize the same version of the encoding template. This is accomplished as follows:

- Backward compatibility – In this case, the message recipient has an older version of the template which contains fewer messages and fields than a newer version used by the sender. The recipient GPB decoder will drop the unknown fields.
- Forward compatibility – The message recipient has a newer version of the template which contains more messages and fields than an older version used by the sender. The recipient GPB decoder will set the values of the missing fields to their default values.

Protobuf has strong support for message evolution. Fields may be added. Optional fields may be upgraded to repeating fields. Fields may also be deleted, as long as their tags are not reused. Enumerations may be extended; when the value of the enumeration is unknown to the decoder, it will drop the enumeration as an unknown field. For a full description of the do's and don'ts of protobuf message evolution, see "Updating A Message Type" in the Protobuf Language Guide.
One important consideration relates to default values. Since default values are applied by the decoder based on its version of the proto file, changes to default values by the message sender will not be known to recipients

with an earlier version of the proto file. When such a change is necessary, the actual [new] value should be set by the sender in all cases, thus overriding the recipient's [old] default setting.

5.3 Message Evolution

In designing for message reuse, it is desirable to minimize the breaking changes which would require redistribution of proto files. This can be done by anticipating the evolving needs to which the protocol may need to adapt. For example:

- Metrics may need to be calculated with additional digits of precision.
- Timestamp may need to be reported with more fine-grained precision (e.g. microseconds instead of milliseconds).
- Formerly unsigned price fields may need to support trading of negative points.
- Dollar (whole number) prices may need to support trading to the penny or fractional penny.
- USD may be replaced with other currencies with different orders of magnitude.

With small tradeoffs in message size (electing signed rather than unsigned, fixed64 rather than fixed32, dynamic exponents rather than static exponents), many of these breaking changes can be avoided.

When it is not possible to avoid a protobuf breaking change, a new .proto version must be issued. The version number will be specified in the GPB Encoding Header.

5.4 Direct Access

High-performance applications may wish to implement "direct access" to selective fields in a GPB message, without incurring the processing overhead of message parsing.

This can be implemented with the following restrictions:

1. Direct access fields must be present in all messages.
2. All "direct access" fields should appear at the front of the GPB message, before any optional fields. Field numbers of direct access fields should be in incremental order and should be lower in value than the field numbers of any non-direct-access fields.
3. Direct access fields should utilize the fixed-length GPB data types, such as fixed32 and fixed64, rather than the variable length encoding data types such as uint32 and uint64.
4. Repeating groups with a variable number of repetitions may not be direct access fields.
5. Variable length strings may not be direct access fields.

This approach supports a hybrid approach to the tradeoffs between message flexibility and performance optimization. A single message definition may contain both critical high-performance fields which support direct-access, while less frequently occurring fields can utilize the full flexibility afforded by GPB.

Note that while the Google-provided protobuf implementations serialize fields in tag number order, this is not a requirement of the specification. Users of non-Google provided implementations who depend on consistent field ordering should verify that their implementations honor with this ordering policy.

5.5 Routing

Protobuf messages can be routed through middlemen who are aware of only a subset of the template with no loss of information. Intermediate servers that don't need to inspect all the data can pass the data through without knowing about all the specific fields.

For example, a simplified proto definition can be created containing only the FIX header, or only select routing fields such as MarketID or SecurityType. An intermediary application using this proto file would parse only a subset of the message, extracting only the fields necessary for routing, but could still forward the entire message payload to the appropriate target application with no loss of data.

This mechanism contributes to the speed of the routing application, and also makes the intermediary system resilient to changes in application-level features in which it has no interest.

5.6 Self-describing Messages

See https://developers.google.com/protocol-buffers/docs/techniques#self-description

6 Message Headers

6.1 Framing for Message Streams

When multiple messages are serialized in a stream, it is necessary to detect where one message ends and the next begins. The Protocol Buffer wire format is not self-delimiting, so the protocol buffer parsers cannot themselves determine where a message ends. This is a common need of all the FPL binary encodings, and as such, it has been addressed by the FPL Simple Open Framing Header standard. This standard defines a normative fixed-length message prefix which identifies the message length and encoding type.

Inclusion of the encoding type in the message frame satisfies three purposes:

1. Supports interleaving of messages of different encodings in a single FIX session.
2. Allows third-party tools such as network protocol analyzers and heterogeneous communication gateways to decode the message without prior knowledge of the message encoding.
3. Provides a version-aware mechanism for evolution of the binary encoding standards. Current GPB identifier reflects version 1.0 of FIX over GPB. New releases of GPB encoding, or new versions of the GPB encoding header would be supported by additional encoding identifier values.

The Simple Open Framing Header is encoded using unsigned binary integer values in Network Byte Order.

Fields:

1. Message length - 32 bit integer containing the total number of bytes of the message, inclusive of the length of the Simple Open Framing Header.
2. Encoding type – 16-bit enumerated value, used to identify the payload encoding type. For the current version of FIX GPB, the encoding type has been set to

FIX GPB v1.0	0x4700

Note that the message length field represents the total message size, including all headers. Since both the Open Framing Header (48 bits) and the GPB Encoding Header (64 bits) are fixed-length, the length of the GPB payload can be easily deduced.

6.2 GPB Encoding Header

The GPB Encoding Header immediately follows the Simple Open Framing Header. Its purpose is to provide the GPB-specific values which are necessary to parse the GPB payload. The three field of the GPB Encoding Header are the proto identifier, proto version and message type.

The GPB Encoding Header is not encoded using Google Protocol Buffers, but rather as a sequence of fixed-length big-endian (network byte order) unsigned binary integers.

The fields of the FIX Protobuf message header are:

- **Proto Id –** a 16-bit integer used to identify the GPB schema.
- **Proto Version Number –** a 16-bit integer used to identify the schema version. The 16-bit number may be further subdivided to reflect the major and minor revisions to the schema.

- **MsgType**—a 32-bit integer or 4-character ASCII code that specifies the GPB message type.

6.2.1 GPB Encoding Header schema

The header fields precede the message body of each GPB message in a fixed layout. Each of these fields shall be encoded as an unsigned integer. The message header is encoded in big-endian byte order. All fields are required.

Header Field	Byte Size	Position Offset
Proto Id	2	0
Proto Version Number	2	2
GPB Message Type	4	4

The total size of the GPB Encoding header is fixed at 64 bits.

6.2.2 Proto Id

This integer identifier identifies the GPB schema (proto file) to be used to decode the message. The proto identifier shall be disseminated with the proto file. The Proto Id may represent any top level GPB schema file or grouping of schema files. If the proto file has been auto-generated from a FIX repository, then the Proto Id will typically represent the subset of the FIX repository reflected by this schema.

6.2.3 Proto Version Number

The Proto Version Number represents the version of the proto schema with which the message was encoded. This version number can be split into major and minor version parts. A minor revision represents a non-breaking change (such as field additions) which may be optionally adopted message decoders, while a major revision represents a breaking change which can only be decoded with the corresponding proto update.

6.2.4 Message Type

This field represents the GPB message. This 32 bit field may reflect a 4 character ASCII identifier, or any integer up to the value of 2^{32}. When the GPB message is derived from a FIX message, the message type field may be the ASCII code of the FIX message, as defined in the FIX repository, for example “AE” for TradeCaptureReport. Alternatively, the message type may reflect the integer enumerated values of the **msg_type** enumeration, as defined in the Session.proto GPB schema.

6.3 Alternate Framing Mechanisms

6.3.1 Alternate Transport Mechanisms

Higher-level transport protocols, which are layered over TCP, often provide their own framing mechanism for delimiting messages and exposing message metadata. In these cases, these mechanisms may be used in place of the FPL Framing Header and GPB fixed-length encoding header to expose GPB meta-data.

Users of these alternate transport protocols, such as message buses, should consider which of the five metadata elements exposed by the FPL Framing Header and GPB Encoder Header need be exposed to their consumers, and should define custom message properties/headers to support these:

a. Message length

b. Encoding Type

c. Proto identifier

d. Proto version

e. Message Type

Examples would be Custom JMS Properties or HTTP Custom Header Properties.

6.3.2 JMS Properties

GPB messages which are transported over a JMS compliant messaging system need not expose the message length to message consumers. Other header fields may be populated in a JMS standard or custom message Properties. Suggested JMS Properties are as follows:

Metadata	**JMS Property**
Encoding Type	FplEncodingType
Proto Id	GpbProtoID
Proto Version Number	GpbProtoVersion
Message Type	GpbMsgType

When these JMS Properties are utilized, the message body need only contain the GBP message payload.

6.3.3 Message Container

As an alternative to the fixed size Encoding Header, a generated GPB **Message Container** can be used to describe and identify messages. The **Message Container** is a generated GPB message that includes the fields from the GPB encoding header, plus an optional field for each of the top-level messages in the repository. When sending a message, the user would populate the GPB encoding header fields in the GPB generated object, and then go on to set the selected optional message field with its message data. Optional fields that are not set are ignored by the serializer.

The advantage of the Message Container is that it simplifies message processing, avoiding the need to process a separate fixed size header to identify the message. Serialization or deserialization can be performed in a single pass. Processing to identify the message is performed on the message container data by identifying the relevant field in the container that corresponds to the message type.

However, as message metadata is contained within the message, the ability to inspect the proto id or version number prior to message parsing is lost. Message containers are thus appropriate for quick prototype or temporal efforts. However, for industrial-strength version-aware parsing, utilization of the FPL message frame and GPB encoding headers is recommended.

A sample Message Container is shown below:

```
import "session-enums.proto";
import "business-message-reject.proto";
import "dont-know-trade.proto";
import "execution-report.proto";
import "heartbeat.proto";
::    ::
import "reject.proto";
import "rfq-request.proto";
import "security-list-request.proto";
import "security-list.proto";
import "test-request.proto";

message MessageContainer {
  // message frame fields
  required uint16 proto_version_no = 3;
  required uint32 msg_type = 4;

  // one optional field for each defined message
  // only one nested message will be populated corresponding to the above msg_type
  optional BusinessReject.BusinessMessageReject business_message_reject = 114;
  optional SingleGeneralOrderHandling.DontKnowTrade dont_know_trade = 115;
  optional SingleGeneralOrderHandling.ExecutionReport execution_report = 116;
  optional Session.Heartbeat heartbeat = 117;
  optional Session.Logon logon = 118;
  optional Session.Logout logout = 119;
}
```


7 Performance Considerations

7.1 Binary Data Representation

The most significant performance advantage of GPB encoding over the existing FIX Tag-Value or XML encodings is due to the inherent efficiency of binary data when compared to string-based data representations.

These efficiencies can be compounded when applications utilize these binary representations not only for the wire encoding associated with serialization, but also for their internal data representation. For example, a timestamp representation as an integral number of milliseconds since 1970 is faster to encode than its associated utcDateTime string. Moreover, a timestamp which is referenced as a binary integer throughout the application can utilize the system's native integer operators to compare timestamps or calculate the difference between them. Furthermore, such an application will also avoid the transformation overhead of mapping between the integer and string representations for data exchange between components.

Greatest performance optimization will be attained when the internal representation of data is aligned with its wire representation.

Data Representation Recommendations:

1. Opt for numeric identifiers for fields such as Account, PartyId, OrderID, ExecID, MDReqID, SecurityID, etc. Set the FIX Repository's isNumeric encoding attribute to TRUE, to instruct the GPB mapping application to generate an integer GPB field.
2. For fields which have a pre-determined set of values, chose GPB enumerations over integers or strings.
3. In places where an ASCII identifier is required, consider mapping the ASCII string to an internal numeric identifier for intra-application use. Map the integer back to the ASCII string for public consumption. For example, map an instrument's Symbol to an integral Security ID for internal communication.
4. Represent dates as the integer number of days since an epoch of choice, for example 01/01/1970 for Maturity Dates and 01/01/1900 for Date of Birth. Provide the epoch as metadata on the proto file via a custom field option. Consuming applications can reference the epoch metadata to repopulate the date as a familiar date string for human-facing interfaces.
5. Represent UTC timestamps as microseconds/milliseconds/seconds since a time epoch. Provide the epoch and time interval as metadata on the proto file via custom field options.
6. Represent decimalized prices as integers. When the precision is a constant value, set the precision as metadata on the proto price field. Otherwise, when the value of the exponent is dynamically determined, create an auxiliary variable to carry the exponent value.

7.2 VarInts vs. Fixed Length Fields

GPB provides two alternate integer encoding mechanisms – varints and fixed-length integers. Varints are "variable length integers", where the length of the field's wire representation is dynamically determined by its value, rather than by its data definition. Varint types are int32, int64, uint32, uint64, sint32, and sint64. Varints result in more compressed serializations, as unused bytes are not populated. Varints are especially useful when

the size of a field varies widely, as there is no loss of bandwidth in defining a data field as a 64 bit varint rather than as a 32 bit integer.

However, empirical testing has shown that serialization of varints is more time-consuming than serialization of the corresponding fixed length integers – fixed32, fixed 64, sfixed32, and sfixed64. It is therefore up to each installation to determine the more significant benefit:

- For optimized compression – elect GPB varints
- For optimized speed – elect GPB fixed length integers

As the stated goal of the HPWG is to optimize for speed, the FIX Repository Auto-Mapping Specification [maps FIX integers into GPB fixed-length data types.

8 Sample Messages

The following listing shows a GPB message definition for a FIX OrderCancelRequest message, its nested components and enumeration types. For simplicity, many of the optional FIX fields and GPB custom options have been omitted.

```
//
//          FIX Unified Repository mapping to Google Protocol Buffer
//
//          Copyright (c) FIX Protocol Ltd. All Rights Reserved.
//
//          File: meta.proto
//
import "google/protobuf/descriptor.proto";

package meta;

enum TimeUnit {
          TIME_UNIT_DAYS = 0;
          TIME_UNIT_SECONDS = 1;
          TIME_UNIT_MILLISECONDS = 2;
          TIME_UNIT_MICROSECONDS = 3;
          TIME_UNIT_NANOSECONDS = 4;
          TIME_UNIT_PICOSECONDS = 5;
}

enum Epoch {
          EPOCH_MIDNIGHT = 0;       // For time-only fields, midnight of a given date
          EPOCH_UNIX = 1;           // Midnight of Jan-01 1970
          EPOCH_1900 = 2;           // Midnight of Jan-01 1900
          EPOCH_2000 = 3;           // Midnight of Jan-01 2000
}

extend google.protobuf.FieldOptions {
          optional TimeUnit time_unit = 51001;
          optional sfixed32 exponent = 51003;
          optional fixed32 min_length = 51004;
          optional fixed32 max_length = 51005;
          optional sfixed64 min_value = 51006;
          optional sfixed64 max_value = 51007;

}

//
//          FIX Unified Repository mapping to Google Protocol Buffer
//
//          Copyright (c) FIX Protocol Ltd. All Rights Reserved.
//
//          File: fix.proto
```

```
//
import "google/protobuf/descriptor.proto";

package fix;

extend google.protobuf.FileOptions {
	optional string category = 53002;
}

extend google.protobuf.MessageOptions {
	optional string msg_type = 55001;
}

extend google.protobuf.FieldOptions {
	optional fixed32 tag = 56003;
	optional Datatype type = 56004;
	optional Version field_added = 56005;
	optional sfixed32 field_added_ep = 56006;
	optional Version field_deprecated = 56007;
}

extend google.protobuf.EnumValueOptions {
	optional string enum = 72004;
	optional Version enum_added = 72005;
	optional sfixed32 enum_added_ep = 72006;
	optional Version enum_deprecated = 72007;
}

enum Datatype {
	CHAR = 0                        [(fix.enum_added)=FIX_2_7];
	DATA = 1                        [(fix.enum_added)=FIX_2_7];
	FLOAT = 2                       [(fix.enum_added)=FIX_2_7];
	INT = 3                         [(fix.enum_added)=FIX_2_7];
	DAY_OF_MONTH = 4                [(fix.enum_added)=FIX_4_1];
	MONTH_YEAR = 5                  [(fix.enum_added)=FIX_4_1];
	AMT = 6                         [(fix.enum_added)=FIX_4_2];
	BOOLEAN = 7                     [(fix.enum_added)=FIX_4_2];
	CURRENCY = 8                    [(fix.enum_added)=FIX_4_2];
	EXCHANGE = 9                    [(fix.enum_added)=FIX_4_2];
	LOCAL_MKT_DATE = 10             [(fix.enum_added)=FIX_4_2];
	MULTIPLE_STRING_VALUE = 11      [(fix.enum_added)=FIX_4_2];
	PRICE = 12                      [(fix.enum_added)=FIX_4_2];
	PRICE_OFFSET = 13               [(fix.enum_added)=FIX_4_2];
	QTY = 14                        [(fix.enum_added)=FIX_4_2];
	STRING = 15                     [(fix.enum_added)=FIX_4_2];
	UTC_TIME_ONLY = 16              [(fix.enum_added)=FIX_4_2];
	UTC_TIMESTAMP = 17              [(fix.enum_added)=FIX_4_2];
	LENGTH = 18                     [(fix.enum_added)=FIX_4_3];
	NUM_IN_GROUP = 19               [(fix.enum_added)=FIX_4_3];
	PERCENTAGE = 20                 [(fix.enum_added)=FIX_4_3];
```



```
        SEQ_NUM = 21                    [(fix.enum_added)=FIX_4_3];
        TAG_NUM = 22                    [(fix.enum_added)=FIX_4_3];
        COUNTRY = 23                    [(fix.enum_added)=FIX_4_4];
        MULTIPLE_CHAR_VALUE = 24        [(fix.enum_added)=FIX_4_4];
        PATTERN = 25                    [(fix.enum_added)=FIX_4_4];
        RESERVED1000PLUS = 26           [(fix.enum_added)=FIX_4_4];
        RESERVED100PLUS = 27            [(fix.enum_added)=FIX_4_4];
        RESERVED4000PLUS = 28           [(fix.enum_added)=FIX_4_4];
        TZ_TIME_ONLY = 29               [(fix.enum_added)=FIX_4_4];
        TZ_TIMESTAMP = 30               [(fix.enum_added)=FIX_4_4];
        TENOR = 31                      [(fix.enum_added)=FIX_4_4];
        UTC_DATE_ONLY = 32              [(fix.enum_added)=FIX_4_4];
        XML_DATA = 33                   [(fix.enum_added)=FIX_5_0];
        LANGUAGE = 34                   [(fix.enum_added)=FIX_5_0_SP_1, (fix.enum_added_ep)=90];
}

enum Version {
        FIX_2_7 = 0;
        FIX_3_0 = 1;
        FIX_4_0 = 2;
        FIX_4_1 = 3;
        FIX_4_2 = 4;
        FIX_4_3 = 5;
        FIX_4_4 = 6;
        FIX_5_0 = 7;
        FIXT_1_1 = 8;
        FIX_5_0_SP_1 = 9;
        FIX_5_0_SP_2 = 10;
}

message Tenor {
        optional fixed32 days = 1;
        optional fixed32 weeks = 2;
        optional fixed32 months = 3;
        optional fixed32 years = 4;
}

//
//      FIX Unified Repository mapping to Google Protocol Buffer
//
//      Copyright (c) FIX Protocol Ltd. All Rights Reserved.
//
//      Category: Session
//
//      File: session.proto
//
import "meta.proto";
import "fix.proto";

option java_outer_classname = "Session";
```

```
option java_package = "org.fixprotocol.components";
package Session;

message HopGrp {
	optional string hop_comp_id = 1 [(fix.tag)=628, (fix.type)=STRING];
	optional fixed32 hop_ref_id = 2 [(fix.tag)=630, (fix.type)=SEQ_NUM];
	optional sfixed64 hop_sending_time = 3 [(fix.tag)=629, (fix.type)=UTC_TIMESTAMP];
}

enum MsgTypeEnum {
	MSG_TYPE_ADJUSTED_POSITION_REPORT = 0 [(fix.enum)="BL"];
	MSG_TYPE_ADVERTISEMENT = 1 [(fix.enum)="7"];
	MSG_TYPE_ALLOCATION_INSTRUCTION = 2 [(fix.enum)="J"];
	MSG_TYPE_ALLOCATION_INSTRUCTION_ACK = 3 [(fix.enum)="P"];
	MSG_TYPE_ALLOCATION_INSTRUCTION_ALERT = 4 [(fix.enum)="BM"];
	MSG_TYPE_ALLOCATION_REPORT = 5 [(fix.enum)="AS"];
	MSG_TYPE_ALLOCATION_REPORT_ACK = 6 [(fix.enum)="AT"];
	MSG_TYPE_APPLICATION_MESSAGE_REPORT = 7 [(fix.enum)="BY"];
	MSG_TYPE_APPLICATION_MESSAGE_REQUEST = 8 [(fix.enum)="BW"];
	MSG_TYPE_APPLICATION_MESSAGE_REQUEST_ACK = 9 [(fix.enum)="BX"];
	MSG_TYPE_ASSIGNMENT_REPORT = 10 [(fix.enum)="AW"];
	MSG_TYPE_BID_REQUEST = 11 [(fix.enum)="k"];
	MSG_TYPE_BID_RESPONSE = 12 [(fix.enum)="l"];
	MSG_TYPE_BUSINESS_MESSAGE_REJECT = 13 [(fix.enum)="j"];
	MSG_TYPE_COLLATERAL_ASSIGNMENT = 14 [(fix.enum)="AY"];
	MSG_TYPE_COLLATERAL_INQUIRY = 15 [(fix.enum)="BB"];
	MSG_TYPE_COLLATERAL_INQUIRY_ACK = 16 [(fix.enum)="BG"];
	MSG_TYPE_COLLATERAL_REPORT = 17 [(fix.enum)="BA"];
	MSG_TYPE_COLLATERAL_REQUEST = 18 [(fix.enum)="AX"];
	MSG_TYPE_COLLATERAL_RESPONSE = 19 [(fix.enum)="AZ"];
	MSG_TYPE_CONFIRMATION = 20 [(fix.enum)="AK"];
	MSG_TYPE_CONFIRMATION_ACK = 21 [(fix.enum)="AU"];
	MSG_TYPE_CONFIRMATION_REQUEST = 22 [(fix.enum)="BH"];
	MSG_TYPE_CONTRARY_INTENTION_REPORT = 23 [(fix.enum)="BO"];
	MSG_TYPE_CROSS_ORDER_CANCEL_REPLACE_REQUEST = 24 [(fix.enum)="t"];
	MSG_TYPE_CROSS_ORDER_CANCEL_REQUEST = 25 [(fix.enum)="u"];
	MSG_TYPE_DERIVATIVE_SECURITY_LIST = 26 [(fix.enum)="AA"];
	MSG_TYPE_DERIVATIVE_SECURITY_LIST_REQUEST = 27 [(fix.enum)="z"];
	MSG_TYPE_DERIVATIVE_SECURITY_LIST_UPDATE_REPORT = 28 [(fix.enum)="BR"];
	MSG_TYPE_DONT_KNOW_TRADE = 29 [(fix.enum)="Q"];
	MSG_TYPE_EMAIL = 30 [(fix.enum)="C"];
	MSG_TYPE_EXECUTION_ACKNOWLEDGEMENT = 31 [(fix.enum)="BN"];
	MSG_TYPE_EXECUTION_REPORT = 32 [(fix.enum)="8"];
	MSG_TYPE_HEARTBEAT = 33 [(fix.enum)="0"];
	MSG_TYPE_IOI = 34 [(fix.enum)="6"];
	MSG_TYPE_LIST_CANCEL_REQUEST = 35 [(fix.enum)="K"];
	MSG_TYPE_LIST_EXECUTE = 36 [(fix.enum)="L"];
	MSG_TYPE_LIST_STATUS = 37 [(fix.enum)="N"];
	MSG_TYPE_LIST_STATUS_REQUEST = 38 [(fix.enum)="M"];
	MSG_TYPE_LIST_STRIKE_PRICE = 39 [(fix.enum)="m"];
```



```
MSG_TYPE_LOGON = 40                                                    [(fix.enum)="A"];
MSG_TYPE_LOGOUT = 41                                                   [(fix.enum)="5"];
MSG_TYPE_MARKET_DATA_INCREMENTAL_REFRESH = 42                          [(fix.enum)="X"];
MSG_TYPE_MARKET_DATA_REQUEST = 43                                      [(fix.enum)="V"];
MSG_TYPE_MARKET_DATA_REQUEST_REJECT = 44                               [(fix.enum)="Y"];
MSG_TYPE_MARKET_DATA_SNAPSHOT_FULL_REFRESH = 45                        [(fix.enum)="W"];
MSG_TYPE_MARKET_DEFINITION = 46                                        [(fix.enum)="BU"];
MSG_TYPE_MARKET_DEFINITION_REQUEST = 47                                [(fix.enum)="BT"];
MSG_TYPE_MARKET_DEFINITION_UPDATE_REPORT = 48                          [(fix.enum)="BV"];
MSG_TYPE_MASS_QUOTE = 49                                               [(fix.enum)="i"];
MSG_TYPE_MASS_QUOTE_ACKNOWLEDGEMENT = 50                               [(fix.enum)="b"];
MSG_TYPE_MULTILEG_ORDER_CANCEL_REPLACE = 51                            [(fix.enum)="AC"];
MSG_TYPE_NETWORK_COUNTERPARTY_SYSTEM_STATUS_REQUEST = 52        [(fix.enum)="BC"];
MSG_TYPE_NETWORK_COUNTERPARTY_SYSTEM_STATUS_RESPONSE = 53       [(fix.enum)="BD"];
MSG_TYPE_NEW_ORDER_CROSS = 54                                          [(fix.enum)="s"];
MSG_TYPE_NEW_ORDER_LIST = 55                                           [(fix.enum)="E"];
MSG_TYPE_NEW_ORDER_MULTILEG = 56                                       [(fix.enum)="AB"];
MSG_TYPE_NEW_ORDER_SINGLE = 57                                         [(fix.enum)="D"];
MSG_TYPE_NEWS = 58                                                     [(fix.enum)="B"];
MSG_TYPE_ORDER_CANCEL_REJECT = 59                                      [(fix.enum)="9"];
MSG_TYPE_ORDER_CANCEL_REPLACE_REQUEST = 60                             [(fix.enum)="G"];
MSG_TYPE_ORDER_CANCEL_REQUEST = 61                                     [(fix.enum)="F"];
MSG_TYPE_ORDER_MASS_ACTION_REPORT = 62                                 [(fix.enum)="BZ"];
MSG_TYPE_ORDER_MASS_ACTION_REQUEST = 63                                [(fix.enum)="CA"];
MSG_TYPE_ORDER_MASS_CANCEL_REPORT = 64                                 [(fix.enum)="r"];
MSG_TYPE_ORDER_MASS_CANCEL_REQUEST = 65                                [(fix.enum)="q"];
MSG_TYPE_ORDER_MASS_STATUS_REQUEST = 66                                [(fix.enum)="AF"];
MSG_TYPE_ORDER_STATUS_REQUEST = 67                                     [(fix.enum)="H"];
MSG_TYPE_POSITION_MAINTENANCE_REPORT = 68                              [(fix.enum)="AM"];
MSG_TYPE_POSITION_MAINTENANCE_REQUEST = 69                             [(fix.enum)="AL"];
MSG_TYPE_POSITION_REPORT = 70                                          [(fix.enum)="AP"];
MSG_TYPE_QUOTE = 71                                                    [(fix.enum)="S"];
MSG_TYPE_QUOTE_CANCEL = 72                                             [(fix.enum)="Z"];
MSG_TYPE_QUOTE_REQUEST = 73                                            [(fix.enum)="R"];
MSG_TYPE_QUOTE_REQUEST_REJECT = 74                                     [(fix.enum)="AG"];
MSG_TYPE_QUOTE_RESPONSE = 75                                           [(fix.enum)="AJ"];
MSG_TYPE_QUOTE_STATUS_REPORT = 76                                      [(fix.enum)="AI"];
MSG_TYPE_QUOTE_STATUS_REQUEST = 77                                     [(fix.enum)="a"];
MSG_TYPE_RFQ_REQUEST = 78                                              [(fix.enum)="AH"];
MSG_TYPE_REGISTRATION_INSTRUCTIONS = 79                                [(fix.enum)="o"];
MSG_TYPE_REGISTRATION_INSTRUCTIONS_RESPONSE = 80                       [(fix.enum)="p"];
MSG_TYPE_REJECT = 81                                                   [(fix.enum)="3"];
MSG_TYPE_REQUEST_FOR_POSITIONS = 82                                    [(fix.enum)="AN"];
MSG_TYPE_REQUEST_FOR_POSITIONS_ACK = 83                                [(fix.enum)="AO"];
MSG_TYPE_RESEND_REQUEST = 84                                           [(fix.enum)="2"];
MSG_TYPE_SECURITY_DEFINITION = 85                                      [(fix.enum)="d"];
MSG_TYPE_SECURITY_DEFINITION_REQUEST = 86                              [(fix.enum)="c"];
MSG_TYPE_SECURITY_DEFINITION_UPDATE_REPORT = 87                        [(fix.enum)="BP"];
MSG_TYPE_SECURITY_LIST = 88                                            [(fix.enum)="y"];
MSG_TYPE_SECURITY_LIST_REQUEST = 89                                    [(fix.enum)="x"];
```



```
        MSG_TYPE_SECURITY_LIST_UPDATE_REPORT = 90                [(fix.enum)="BK"];
        MSG_TYPE_SECURITY_STATUS = 91                            [(fix.enum)="f"];
        MSG_TYPE_SECURITY_STATUS_REQUEST = 92                    [(fix.enum)="e"];
        MSG_TYPE_SECURITY_TYPE_REQUEST = 93                      [(fix.enum)="v"];
        MSG_TYPE_SECURITY_TYPES = 94                             [(fix.enum)="w"];
        MSG_TYPE_SEQUENCE_RESET = 95                             [(fix.enum)="4"];
        MSG_TYPE_SETTLEMENT_INSTRUCTION_REQUEST = 96             [(fix.enum)="AV"];
        MSG_TYPE_SETTLEMENT_INSTRUCTIONS = 97                    [(fix.enum)="T"];
        MSG_TYPE_SETTLEMENT_OBLIGATION_REPORT = 98               [(fix.enum)="BQ"];
        MSG_TYPE_STREAM_ASSIGNMENT_REPORT = 99                   [(fix.enum)="CD"];
        MSG_TYPE_STREAM_ASSIGNMENT_REPORT_ACK = 100              [(fix.enum)="CE"];
        MSG_TYPE_STREAM_ASSIGNMENT_REQUEST = 101                 [(fix.enum)="CC"];
        MSG_TYPE_TEST_REQUEST = 102                              [(fix.enum)="1"];
        MSG_TYPE_TRADE_CAPTURE_REPORT = 103                      [(fix.enum)="AE"];
        MSG_TYPE_TRADE_CAPTURE_REPORT_ACK = 104                  [(fix.enum)="AR"];
        MSG_TYPE_TRADE_CAPTURE_REPORT_REQUEST = 105              [(fix.enum)="AD"];
        MSG_TYPE_TRADE_CAPTURE_REPORT_REQUEST_ACK = 106          [(fix.enum)="AQ"];
        MSG_TYPE_TRADING_SESSION_LIST = 107                      [(fix.enum)="BJ"];
        MSG_TYPE_TRADING_SESSION_LIST_REQUEST = 108              [(fix.enum)="BI"];
        MSG_TYPE_TRADING_SESSION_LIST_UPDATE_REPORT = 109        [(fix.enum)="BS"];
        MSG_TYPE_TRADING_SESSION_STATUS = 110                    [(fix.enum)="h"];
        MSG_TYPE_TRADING_SESSION_STATUS_REQUEST = 111            [(fix.enum)="g"];
        MSG_TYPE_USER_NOTIFICATION = 112                         [(fix.enum)="CB"];
        MSG_TYPE_USER_REQUEST = 113                              [(fix.enum)="BE"];
        MSG_TYPE_USER_RESPONSE = 114                             [(fix.enum)="BF"];
        MSG_TYPE_XMLNON_FIX = 115                                [(fix.enum)="n"];
}

enum ApplVerIdEnum {
        APPL_VER_ID_FIX27 = 0                                    [(fix.enum)="0"];
        APPL_VER_ID_FIX30 = 1                                    [(fix.enum)="1"];
        APPL_VER_ID_FIX40 = 2                                    [(fix.enum)="2"];
        APPL_VER_ID_FIX41 = 3                                    [(fix.enum)="3"];
        APPL_VER_ID_FIX42 = 4                                    [(fix.enum)="4"];
        APPL_VER_ID_FIX43 = 5                                    [(fix.enum)="5"];
        APPL_VER_ID_FIX44 = 6                                    [(fix.enum)="6"];
        APPL_VER_ID_FIX50 = 7                                    [(fix.enum)="7"];
        APPL_VER_ID_FIX50SP1 = 8                                 [(fix.enum)="8"];
        APPL_VER_ID_FIX50SP2 = 9                                 [(fix.enum)="9"];
}

message StandardHeader {
        optional string deliver_to_comp_id = 1                   [(fix.tag)=128, (fix.type)=STRING];
        optional string deliver_to_sub_id = 2                    [(fix.tag)=129, (fix.type)=STRING];
        optional fixed32 msg_seq_num = 3                         [(fix.tag)=34, (fix.type)=SEQ_NUM];
        optional string on_behalf_of_comp_id = 4                 [(fix.tag)=115, (fix.type)=STRING];
        optional string on_behalf_of_sub_id = 5                  [(fix.tag)=116, (fix.type)=STRING];
        optional sfixed64 orig_sending_time = 6                  [(fix.tag)=122, (fix.type)=UTC_TIMESTAMP];
        optional bool poss_dup_flag = 7                          [(fix.tag)=43, (fix.type)=BOOLEAN];
        optional bool poss_resend = 8                            [(fix.tag)=97, (fix.type)=BOOLEAN];
```



```
	optional bytes secure_data = 9 [(fix.tag)=91, (fix.type)=DATA, (fix.field_deprecated)=FIXT_1_1];
	optional fixed32 secure_data_len = 10 [(fix.tag)=90, (fix.type)=LENGTH, (fix.field_deprecated)=FIXT_1_1];
	optional string sender_comp_id = 11 [(fix.tag)=49, (fix.type)=STRING];
	optional string sender_sub_id = 12 [(fix.tag)=50, (fix.type)=STRING];
	optional sfixed64 sending_time = 13 [(fix.tag)=52, (fix.type)=UTC_TIMESTAMP];
	optional string target_comp_id = 14 [(fix.tag)=56, (fix.type)=STRING];
	optional string target_sub_id = 15 [(fix.tag)=57, (fix.type)=STRING];
	optional string deliver_to_location_id = 16 [(fix.tag)=145, (fix.type)=STRING];
	optional string on_behalf_of_location_id = 17 [(fix.tag)=144, (fix.type)=STRING];
	optional string sender_location_id = 18 [(fix.tag)=142, (fix.type)=STRING];
	optional string target_location_id = 19 [(fix.tag)=143, (fix.type)=STRING];
	optional fixed32 last_msg_seq_num_processed = 20 [(fix.tag)=369, (fix.type)=SEQ_NUM];
	optional string message_encoding = 21 [(fix.tag)=347, (fix.type)=STRING];
	optional bytes xml_data = 22 [(fix.tag)=213, (fix.type)=DATA];
	optional fixed32 xml_data_len = 23 [(fix.tag)=212, (fix.type)=LENGTH];
	optional ApplVerIdEnum appl_ver_id = 24 [(fix.tag)=1128, (fix.type)=STRING];
	optional string cstm_appl_ver_id = 25 [(fix.tag)=1129, (fix.type)=STRING];
	repeated HopGrp hop_grp = 26 [(fix.tag)=627];
	optional sfixed64 appl_ext_id = 27 [(fix.tag)=1156, (fix.type)=INT];
}

message StandardTrailer {
	optional bytes signature = 1 [(fix.tag)=89, (fix.type)=DATA, (fix.field_deprecated)=FIXT_1_1];
	optional fixed32 signature_length = 2 [(fix.tag)=93, (fix.type)=LENGTH, (fix.field_deprecated)=FIXT_1_1];
}

//
//	FIX Unified Repository mapping to Google Protocol Buffer
//
//	Copyright (c) FIX Protocol Ltd. All Rights Reserved.
//
//	Category: SingleGeneralOrderHandling
//
//	File: single-general-order-handling.proto
//
import "meta.proto";
import "fix.proto";
import "session.proto";

option java_outer_classname = "SingleGeneralOrderHandling";
option java_package = "org.fixprotocol.components";
package SingleGeneralOrderHandling;

enum OrdStatusEnum {
	ORD_STATUS_CANCELED = 0 [(fix.enum)="4"];
	ORD_STATUS_DONE_FOR_DAY = 1 [(fix.enum)="3"];
	ORD_STATUS_FILLED = 2 [(fix.enum)="2"];
	ORD_STATUS_NEW = 3 [(fix.enum)="0"];
	ORD_STATUS_PARTIALLY_FILLED = 4 [(fix.enum)="1"];
	ORD_STATUS_PENDING_CANCEL = 5 [(fix.enum)="6"];
```

```
        ORD_STATUS_REJECTED = 6                                 [(fix.enum)="8"];
        ORD_STATUS_STOPPED = 7                                  [(fix.enum)="7"];
        ORD_STATUS_PENDING_NEW = 8                              [(fix.enum)="A"];
        ORD_STATUS_SUSPENDED = 9                                [(fix.enum)="9"];
        ORD_STATUS_CALCULATED = 10                              [(fix.enum)="B"];
        ORD_STATUS_EXPIRED = 11                                 [(fix.enum)="C"];
        ORD_STATUS_ACCEPTED_FOR_BIDDING = 12                    [(fix.enum)="D"];
        ORD_STATUS_PENDING_REPLACE = 13                         [(fix.enum)="E"];
        ORD_STATUS_REPLACED = 14                    [(fix.enum)="5", (fix.enum_deprecated)=FIX_4_3];
}

enum WorkingIndicatorEnum {
        WORKING_INDICATOR_NOT_WORKING = 0                       [(fix.enum)="N"];
        WORKING_INDICATOR_WORKING = 1                           [(fix.enum)="Y"];
}

enum AcctIdSourceEnum {
        ACCT_ID_SOURCE_BIC = 0                                  [(fix.enum)="1"];
        ACCT_ID_SOURCE_DTCCCODE = 1                             [(fix.enum)="5"];
        ACCT_ID_SOURCE_OMGEO = 2                                [(fix.enum)="4"];
        ACCT_ID_SOURCE_OTHER = 3                                [(fix.enum)="99"];
        ACCT_ID_SOURCE_SID_CODE = 4                             [(fix.enum)="2"];
        ACCT_ID_SOURCE_TFM = 5                                  [(fix.enum)="3"];
}

enum AccountTypeEnum {
        ACCOUNT_TYPE_CARRIED_CUSTOMER_SIDE = 0                  [(fix.enum)="1"];
        ACCOUNT_TYPE_CARRIED_NON_CUSTOMER_SIDE = 1              [(fix.enum)="2"];
        ACCOUNT_TYPE_CARRIED_NON_CUSTOMER_SIDE_CROSS_MARGINED = 2       [(fix.enum)="6"];
        ACCOUNT_TYPE_FLOOR_TRADER = 3                           [(fix.enum)="4"];
        ACCOUNT_TYPE_HOUSE_TRADER = 4                           [(fix.enum)="3"];
        ACCOUNT_TYPE_HOUSE_TRADER_CROSS_MARGINED = 5            [(fix.enum)="7"];
        ACCOUNT_TYPE_JOINT_BACK_OFFICE_ACCOUNT = 6              [(fix.enum)="8"];
}

enum CxlRejResponseToEnum {
        CXL_REJ_RESPONSE_TO_ORDER_CANCEL = 0                    [(fix.enum)="2"];
        CXL_REJ_RESPONSE_TO_ORDER_CANCEL_REQUEST = 1            [(fix.enum)="1"];
}

enum CxlRejReasonEnum {
        CXL_REJ_REASON_TOO_LATE_TO_CANCEL = 0                   [(fix.enum)="0"];
        CXL_REJ_REASON_UNKNOWN_ORDER = 1                        [(fix.enum)="1"];
        CXL_REJ_REASON_BROKER_CREDIT = 2                        [(fix.enum)="2"];
        CXL_REJ_REASON_ORDER_ALREADY_IN_PENDING_STATUS = 3      [(fix.enum)="3"];
        CXL_REJ_REASON_DUPLICATE_CL_ORD_ID = 4                  [(fix.enum)="6"];
        CXL_REJ_REASON_ORIG_ORD_MOD_TIME = 5                    [(fix.enum)="5"];
        CXL_REJ_REASON_UNABLE_TO_PROCESS_ORDER_MASS_CANCEL_REQUEST = 6          [(fix.enum)="4"];
        CXL_REJ_REASON_OTHER = 7                                [(fix.enum)="99"];
        CXL_REJ_REASON_INVALID_PRICE_INCREMENT = 8              [(fix.enum)="18"];
```



```
        CXL_REJ_REASON_PRICE_EXCEEDS_CURRENT_PRICE = 9              [(fix.enum)="7"];
        CXL_REJ_REASON_PRICE_EXCEEDS_CURRENT_PRICE_BAND = 10       [(fix.enum)="8"];
}

message AcctIdSourceUnion {
        optional AcctIdSourceEnum acct_id_source = 1;
        optional sfixed64 acct_id_source_sfixed64 = 2;
}

message CxlRejReasonUnion {
        optional CxlRejReasonEnum cxl_rej_reason = 1;
        optional sfixed64 cxl_rej_reason_sfixed64 = 2;
}

message OrderCancelReject {
        option (fix.msg_type) = "9";
        optional string cl_ord_id = 1                                  [(fix.tag)=11, (fix.type)=STRING];
        optional CxlRejReasonUnion cxl_rej_reason = 2                  [(fix.tag)=102, (fix.type)=INT];
        optional string list_id = 3                                    [(fix.tag)=66, (fix.type)=STRING];
        optional string order_id = 4                                   [(fix.tag)=37, (fix.type)=STRING];
        optional Session.StandardHeader standard_header = 5;
        optional Session.StandardTrailer standard_trailer = 6;
        optional string text = 7                                       [(fix.tag)=58, (fix.type)=STRING];
        optional OrdStatusEnum ord_status = 8                          [(fix.tag)=39, (fix.type)=CHAR];
        optional string orig_cl_ord_id = 9                             [(fix.tag)=41, (fix.type)=STRING];
        optional string secondary_order_id = 10                        [(fix.tag)=198, (fix.type)=STRING];
        optional string account = 11                                   [(fix.tag)=1, (fix.type)=STRING];
        optional CxlRejResponseToEnum cxl_rej_response_to = 12 [(fix.tag)=434, (fix.type)=CHAR];
        optional bytes encoded_text = 13                               [(fix.tag)=355, (fix.type)=DATA];
        optional fixed32 encoded_text_len = 14                         [(fix.tag)=354, (fix.type)=LENGTH];
        optional sfixed64 transact_time = 15                           [(fix.tag)=60, (fix.type)=UTC_TIMESTAMP];
        optional AccountTypeEnum account_type = 16                     [(fix.tag)=581, (fix.type)=INT];
        optional string cl_ord_link_id = 17                            [(fix.tag)=583, (fix.type)=STRING];
        optional sfixed64 orig_ord_mod_time = 18                       [(fix.tag)=586, (fix.type)=UTC_TIMESTAMP];
        optional string secondary_cl_ord_id = 19                       [(fix.tag)=526, (fix.type)=STRING];
        optional sfixed32 trade_origination_date = 20                  [(fix.tag)=229, (fix.type)=LOCAL_MKT_DATE];
        optional bool working_indicator = 21                           [(fix.tag)=636, (fix.type)=BOOLEAN];
        optional AcctIdSourceUnion acct_id_source = 22                 [(fix.tag)=660, (fix.type)=INT];
        optional sfixed32 trade_date = 23                              [(fix.tag)=75, (fix.type)=LOCAL_MKT_DATE];
}
```


9 Appendix A – FIX Messages by Category

FIX Section	FIX Category	Proto file name	FIX Messages
N/A	Common	common.proto	common components
Session	Session	session.proto	Heartbeat <0> TestRequest <1> ResendRequest <2> Reject <3> SequenceReset <4> Logout <5> Logon <A> XML_non_FIX <n>
Pre Trade	Indication	indication.proto	IOI <6> Advertisement <7>
	EventCommunication	event-communication.proto	News <B> Email <C>
	QuotationNegotiation	quotation-negotiation.proto	QuoteRequest <R> Quote <S> QuoteCancel <Z> QuoteStatusRequest <a> MassQuoteAcknowledgement <b> MassQuote <i> QuoteRequestReject <AG> RFQRequest <AH> QuoteStatusReport <AI> QuoteResponse <AJ>
	MarketData	market-data.proto	MarketDataRequest <V> MarketDataSnapshotFullRefresh <W> MarketDataIncrementalRefresh <X> MarketDataRequestReject <Y> StreamAssignmentRequest <CC> StreamAssignmentReport <CD>

			StreamAssignmentReportACK <CE>
	SecuritiesReferenceData	securities-reference-data.proto	SecurityDefinitionRequest <c> SecurityDefinition <d> SecurityStatusRequest <e> SecurityStatus <f> SecurityTypeRequest <v> SecurityTypes <w> SecurityListRequest <x> SecurityList <y> DerivativeSecurityListRequest <z> DerivativeSecurityList <AA> SecurityDefinitionUpdateReport <BP> SecurityListUpdateReport <BK> DerivativeSecurityListUpdateReport
	MarketStructureReferenceData	market-structure-reference-data.proto	TradingSessionStatusRequest <g> TradingSessionStatus <h> TradingSessionList <BJ> TradingSessionListRequest <BI> TradingSessionListUpdateReport <BS> MarketDefinitionRequest <BT> MarketDefinition <BU> MarketDefinitionUpdateReport <BV>
	PartiesReferenceData	parties-reference-data.proto	PartyDetailsListRequest <CF> PartyDetailsListReport <CG>
Trade	SingleGeneralOrderHandling	single-general-order-handling.proto	ExecutionReport <8> OrderCancelReject <9> NewOrderSingle <D> OrderCancelRequest <F> OrderCancelReplaceRequest <G> OrderStatusRequest <H> DontKnowTradeDK <Q> ExecutionAcknowledgement <BN>
	ProgramTrading	program-trading.proto	NewOrderList <E> ListCancelRequest <K>

			ListExecute <L> ListStatusRequest <M> ListStatus <N> BidRequest <k> BidResponse <l> ListStrikePrice <m>
	OrderMassHandling	order-mass-handling.proto	OrderMassCancelRequest <q> OrderMassCancelReport <r> OrderMassStatusRequest <AF> OrderMassActionReport <BZ> OrderMassActionRequest <CA>
	CrossOrders	cross-orders.proto	NewOrderCross <s> CrossOrderCancelReplaceRequest <t> CrossOrderCancelRequest <u>
	MultilegOrders	multileg-orders.proto	NewOrderMultileg <AB> MultilegOrderCancelReplace <AC>
Post Trade	Allocation	allocation.proto	AllocationInstruction <J> AllocationInstructionAck <P> AllocationReport <AS> AllocationReportAck <AT> AllocationInstructionAlert <BM>
	SettlementInstruction	settlement-instruction.proto	SettlementInstructions <T> SettlementInstructionRequest <AV> SettlementObligationReport <BQ>
	RegistrationInstruction	registration-instruction.proto	RegistrationInstructions <o> RegistrationInstructionsResponse <p>
	TradeCapture	trade-capture.proto	TradeCaptureReportRequest <AD> TradeCaptureReport <AE> TradeCaptureReportRequestAck <AQ> TradeCaptureReportAck <AR>
	Confirmation	confirmation.proto	Confirmation <AK> Confirmation_Ack <AU> ConfirmationRequest <BH>

	PositionMaintenance	position-maintenance.proto	PositionMaintenanceRequest <AL> PositionMaintenanceReport <AM> RequestForPositions <AN> RequestForPositionsAck <AO> PositionReport <AP> AssignmentReport <AW> ContraryIntentionReport <BO> AdjustedPositionReport <BL>
	CollateralManagement	collateral-management.proto	CollateralRequest <AX> CollateralAssignment <AY> CollateralResponse <AZ> CollateralReport <BA> CollateralInquiry <BB> CollateralInquiryAck <BG>
Infrastructure	BusinessReject	business-reject.proto	BusinessMessageReject <j>
	Network	network.proto	NetworkCounterpartySystemStatusRequest <BC> NetworkCounterpartySystemStatusResponse <BD>
	UserManagement	user-management.proto	UserRequest <BE> UserResponse <BF> UserNotification <CB>
	Application	application.proto	ApplicationMessageRequest <BW> ApplicationMessageRequestAck <BX> ApplicationMessageReport <BY>

