MiFiD II – Clock Synchronization Working Group

Global Technical Committee

TimeStamp Datatypes Enhancements

November 30, 2015

Revision0.8

Proposal Status: Approved~~Draft~~

For Global Technical Committee Governance Internal Use Only

Submission Date	November 30, 2015	Control Number	EP206
Submission Status	Approved	Ratified Date	Jan 21, 2016
Primary Contact Person	Yuval Cohen	Release Identifier	5.0.SP3

r3.2

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 6
1 Introduction 8
1.1 High Resolution Time Data 8
1.2 Survey of FIX Encodings 8
1.3 Enhancement Options 8
1.4 Support Levels 8
1.5 Local Market Date datatype Errata 9
1.6 Inband identification of time precision 9
2 Business Requirements 10
2.1 Existing Datatype that requires changes: 11
2.1.1 FIX datatypes 11
2.1.2 FIXML datatypes 13
2.2 Existing FIX fields that require conveying timestamps with microseconds' granularity: 18
2.2.1 Existing fields of data-type: UTCTimestamp 18
3 Issues and Discussion Points 20
3.1 Implementation alternatives 20
3.1.1 Implementation alternatives analysis 21
3.1.2 Backwards compatibility analysis 22
3.2 Maximum Precision 22
3.2.1 Number Precision 22
3.3 References external discussions: 22
3.3.1 Timestamp discussions: 22
3.3.2 LocalMktDate discussions: 22
3.4 Existing Implementation of higher precisions 23
4 Recommended Approach 23
4.1 Time Precision Conservation Principle 23
4.2 Maximum Time Precision 24
4.3 Recommendations on decimal increments 24
4.4 Recommendations on trimming 24
4.5 Recommendations on padding 24
5 Proposed Message Flow 24
6 FIX Message Tables 25
7 FIX Component Blocks 25
8 Category Changes 25
9 FIX Specification Errata 25
Appendix A - Data Dictionary 26
Appendix B - Glossary Entries 27
Appendix C - Abbreviations 27

Appendix D - Usage Examples 27

Table of Figures

No table of figures entries found.

Document History

Revision	Date	Author	Revision Comments
r0.1	15/10/2015	Yuval Cohen	Initial draft - alternatives.
r0.2	15/10/2015	Jim Northey	Added Support levels section and drafting corrections
r0.3	15/10/2015	Neil Horlock	Minor tweaks to drafting.
R0.4	16/10/2015	Jim Northey	Added references to existing implementation, added discussion around merits for a maximum precision, added a field to indicate time precision.
R0.5	19/10/2015	Neil Horlock	Added Metatype option for proposal 1c
R0.6	20/10/2015	Yuval Cohen	Refined the number of precision Summarize options Added a discussion about the number of precision Added existing implementations
R0.7	19/11/2015	Jim Northey	Add discussion on existing support by FIX for extended time precision, such as FAST, SBE, GPB, ASN.1, and FIXML. Address supporting that timestamps can be <= to the maximum time precision in use in a FIX session. Omit trailing zeroes – which implies a variable length time precision. Add a section on the need to conserve time precision due to technical and operational burden imposed by increased time precision. Added section recommending that specific field precision should be specified in the rules of engagement.
R0.8	30/11/2015	Jim Northey	Included the discussion on Picoseconds drafted by Neil Horlock. Removed the additional fields proposed for FIXT1.2.
	29/12/2015	R. Shriver	ASBUILT created.

Revision	Date	Author	Revision Comments
	20/01/2016	R. Shriver	Revised section 2.1.1 FIX datatypes and added section 2.1.2 FIXML datatypes and revised formatting to align with current conventions for removing and adding text to the datatype table. Cleaned up simple typographical errors.

1 Introduction

1.1 High Resolution Time Data

In the tag=value encodings, the FIX timestamp data-types are defined (and implemented) to support millisecond granularity. In recent years, when HFT (High Frequency Trading) activity expanded, the requirement for timestamps precision and reporting in microseconds and/or nanoseconds granularity emerged. This requirement has now become explicit by MiFiD II regulations. MiFiD II is not the first time that such requirement has been seen; it was raised in a few technical discussions (see references below). Some execution venues have already implemented using different non-standard approaches to convey timestamps in microseconds and/or nanoseconds.

The requirements to enhance the tag=value with higher timestamp resolution(s) is relevant to all supported versions of FIX: FIX4.2, FIX.4.4, FIX 5.0,~~ and~~ FIX.5.0~~1~~SP1, and FIX.5.0SP2.

1.2 Survey of FIX Encodings

FIX currently provides these encodings: FIX Tag=Value (FTV), FIXML, Simple Binary Encoding (SBE), Google Protocol Buffers (GPB), FAST, and ASN.1.
FIXML uses the W3C XML Schema xs:dateT~~t~~ime datatype for timestamps. The xs:dateT~~t~~ime format is ISO 8601 compatible. No change is required to FIXML, as ISO 8601 does not limit the number of decimal places. The FIXML Encoding is also compatible with the MIFID II and MIFIR requirement for ISO 8601 date formats.
FAST, SBE, GPB, FAST, and ASN.1 can all support increased decimal precision for timestamps up to nanosecond precision. Each of these encodings will require enhancement to support picosecond resolution.

1.3 Enhancement Options

There are different technical options that should be considered to enhance the FIX Protocol for higher resolution time. In this document we discuss the following options:

1.a. ISO 8061 format conveyed in existing FIX fields
1.b. ‘FIX like’ format with optional higher resolution conveyed in existing FIX fields
1.c. ‘Meta type’ which supports either ISO 8061 or ‘FIX like’ format conveyed in existing FIX fields
2.a. ISO 8061 format conveyed in new FIX fields
2.b. Precisions only conveyed in new FIX fields

Section 3 shows the analysis and discuss in details the pros and cons of these options.

1.4 Support Levels

The increase in time resolution has the potential to be quite disruptive for the FIX Trading Community. This disruption will be felt most by earlier version of FIX, such as FIX.4.2 and FIX.4.4 that do not contain in band (fields in messages) to indicate extension packs. Also, the expansion of time fields will have an impact on bandwidth consumption and storage requirements for FIX log files. There is also a realization that not all participants in the market require high resolution timing. In this context of a large install base that must continue to operate and not be disrupted as major market infrastructure is being migrated to higher resolution timing, this proposal recommends that all FIX services, via rules of engagements, identify the level of support provided for high resolution time. The levels proposed are:

0. No Support – FIX Service will likely experience a processing exception if high resolution time data is encountered.
1. High Resolution Tolerant – FIX Service can receive higher resolution time data, but will not transmit higher resolution time data and will not guarantee persistence of any higher resolution time data received.
2. Application Level Support for Higher resolution time data – FIX Service supports sending and receiving higher resolution time data and will persist higher resolution time data that is encountered within the application messages. The maximum precision shall be agreed upon out of band by counterparty agreement.
3. Full Support for Higher resolution time data – FIX Service supports sending and receiving higher resolution time data and will persist higher resolution time data at both the session layer and the application layer. The maximum precision shall be agreed upon out of band by counterparty agreement.

Rules of engagement should specify the precision of the decimal second that is supported by the FIX Service. FIX Service providers are encouraged to make the resolution configurable.

1.5 Local Market Date datatype Errata

The local market date format is ‘YYYMMDD’, whilst the FIX repository provides a wrong example:
BizDate=”2003-09-10”
We propose to correct the example of the datatype.

1.6 Inband identification of time precision

An initial p~~P~~roposal was~~is~~ to add~~ a~~ fields for the next version of FIX session layer (FIXT1.2) that ~~will~~ would indicate the number of decimal places that will be used in the UTCTimestamp and UTCTime fields. The addition of new fields ~~should be offset~~was felt unnecessary. The ~~by the ability for a~~ FIX service ~~to~~ can easily detect the precision at both the session layer and the application layer via examination of the time fields, such as SendingTime(52) or TransactTime(60). The use of additional fields to explicitly state time precision inband will lead to additional testing and confusion, where the fields would either be omitted or not correctly represent the actual precision in use.

2 Business Requirements

ESMA (European Securities and Markets Authority) recently published the Regulatory technical and implementation standards – Annex I: (i.e. MiFiD II / MiFiR)

Within these regulations, there are requirements to communicate timesstamps of certain~~s~~ events with granularities of ‘1 microsecond or better’.

The range of timestamps reporters includes execution venues, banks, buy-side customers and others.

The workflows in scope for these regulations include:

- Pre-trade
  - Indications
  - Quotation Negotiation (click to trade)
  - Market data
  - Security Reference data
  - Market Structure Reference Data
- Trade
  - Single General Order Handling
  - Program Trading
  - Order Mass Handling
  - Cross Orders
  - Multileg orders
- Post -trade
  - Trade Capture

As we can see from the above list, most (if not all) of the timestamps events may be in scope.

In order to maintain consistency across all the workflows, we assume that the entire FIX protocol will need to support timestamp events in granularity of 1 microsecond or better.

The scope of the requirements includes all existing supported FIX versions, with the assumption that versions are FIX.4.2,FIX. 4.4 and FIX.5.0SP1, and FIX.5.0SP2.

Although the current business requirements speak~~s~~ about ‘microseconds’, we strongly recommend to enhance the standards to remove any limitation of timestamp granularities.

2.1 Existing Datatype that requires changes:

Proposed changes are highlighted

2.1.1 FIX datatypes

Type Name	Description	Added in FIX version
UTCTimestamp	string field representing tTime/date combination represented in UTC (Universal Time Coordinated, also known as "GMT") in either YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss* (milliseconds) format, colons, dash, and period required. Valid values: * YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second) (without milliseconds). * YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss* fFractions of seconds. The fractions of seconds may be empty when no franctions of seconds are conveyed (in such a case the period is not conveyed), it may include 3 digits to convey milliseconds, 6 digits to convey microseconds, 9 digits to convey nanoseconds, 12 digits to convey picoseconds; Other number of digits may be used with bilateral agreement. (indicating milliseconds). Leap Seconds: Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has never utilized these dates. During a leap second insertion, a UTCTimestamp field may read "19981231-23:59:59", "19981231-23:59:60", "19990101-00:00:00". (see http://tycho.usno.navy.mil/leapsec.html) Example(s) TransactTime(tag 60)="""20011217-09:30:47.123""" millisecond TransactTime(60)="""20011217-09:30:47.123456""" microseconds TransactTime(60)="20011217-09:30:47.123456789" nanoseconds TransactTime(60)="20011217-09:30:47.123456789123" picoseconds	FIX.4.2
UTCTimeOnly	string field representing Ttime-only represented in UTC (Universal Time Coordinated, also known as "GMT") in either HH:MM:SS (whole seconds) or HH:MM:SS.sss* (milliseconds) format, colons, and period required. This special-purpose field is paired with	FIX.4.2

	UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: ~~HH = 00-23, MM = 00-60 (60 only if UTC leap second), SS = 00-59. (without milliseconds)~~ HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss* f~~F~~ractions of seconds. The fractions of seconds may be empty when no fra~~n~~ctions of seconds are conveyed (in such a case the period is not conveyed), it may include 3 digits to convey milliseconds, 6 digits to convey microseconds, 9 digits to convey nanoseconds, 12 digits to convey picoseconds; Other number of digits may be used with bilateral agreement ~~=000-999 (indicating milliseconds).~~ Example(s) MDEntryTime(~~tag~~ 273)="13:20:00.123"milliseconds MDEntryTime(273)="13:20:00.123456" microseconds MDEntryTime(273)="13:20:00.123456789" nanoseconds MDEntryTime(273)="13:20:00.123456789123" picoseconds	
TZTimeOnly	string field representing the time represented based on ISO 8601. This is the time with a UTC offset to allow identification of local time and timezone of that time. Format is HH:MM[:SS][Z | [+ | - hh[:mm]]] where HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, hh = 01-12 offset hours, mm = 00-59 offset minutes. Example(s) ~~Example:~~ "07:39Z" is 07:39 UTC ~~Example:~~ "02:39-05" is five hours behind UTC, thus Eastern Time ~~Example:~~ "15:39+08" is eight hours ahead of UTC, Hong Kong/Singapore time ~~Example:~~ "13:09+05:30" is 5.5 hours ahead of UTC, India time	FIX.4.4 EP-1
TZTimestamp	string field representing a time/date combination representing local time with an offset to UTC to allow identification of local time and timezone offset of that time. The representation is based on ISO 8601. Format is YYYYMMDD-HH:MM:SS.sss*[Z | [+ | - hh[:mm]]] where YYYY = 0000 to 9999, MM = 01-12, DD = 01-31 HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, hh = 01-12 offset hours, mm = 00-59 offset minutes, sss* f~~F~~ractions of seconds. The fractions of seconds may be empty when no fra~~n~~ctions of seconds are conveyed (in such a case the period is not conveyed), it may include 3 digits to convey milliseconds, 6 digits to convey microseconds, 9 digits to convey nanoseconds, 12	FIX.4.4 EP-1

	digits to convey picoseconds; Other number of digits may be used with bilateral agreement Example(s) Example: "20060901-07:39Z" is 07:39 UTC on 1st of September 2006 Example: "20060901-02:39-05" is five hours behind UTC, thus Eastern Time on 1st of September 2006 Example: "20060901-15:39+08" is eight hours ahead of UTC, Hong Kong/Singapore time on 1st of September 2006 Example: "20060901-13:09+05:30" is 5.5 hours ahead of UTC, India time on 1st of September 2006 Using decimal seconds: "20060901-13:09.123+05:30" milliseconds "20060901-13:09.123456+05:30" microseconds "20060901-13:09.123456789+05:30" nanoseconds "20060901-13:09.123456789123+05:30" picoseconds "20060901-13:09.123456789Z" nanoseconds UTC timezone	
LocalMktDate	string field representing a Date of Local Market (as opposed to UTC) in YYYYMMDD format. This is the "normal" date field used by the FIX Protocol. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31. Example(s) BizDate="2003-09-10" Example: MaturityDate(541)="20150724"	FIX.4.2

2.1.2 FIXML datatypes

Type Name	Description	Added in FIX version
UTCTimestamp Base Type: String XML Builtin: 0	string field representing Time/date and time combination represented in UTC (Universal Time Coordinated (UTC), also known as Greenwich Mean Time ("GMT").	FIX.4.2

XML Base: xs:dateTime XML Pattern[RS1]:	Its value space is described as the combination of date and time of day in the Chapter 5.4 of ISO 8601. Valid values are in the format in either YYYY-MM-DDTHH:MM:SS (whole seconds) or YYYY-MM-DDTHH:MM:SS.sss (milliseconds) format as specified in ISO 8601. Valid values: * YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second) (without milliseconds). * where YYYY = 0000-9999 year, MM = 01-12 month, DD = 01-31 day, HH = 00-23 hour, MM = 00-59 minute, SS = 00-60 second (60 only if UTC leap second), and optionally sss (one or more digits representing a decimal fraction of a second). =000-999 (indicating milliseconds). The punctuation of "-", ":" and the string value of "T" to separate the date and time are required. The "." is only required when sub-second time precision is specified. Leap Seconds: Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has never utilized these dates. During a leap second insertion, a UTCTimestamp field may read "1998-12-31T23:59:59", "1998-12-31T23:59:60", "1999-01-01T00:00:00". (see http://tycho.usno.navy.mil/leapsec.html) Example(s) TxnransactTm="2001-12-17T09:30:47-05:00" seconds TxnTm="20011217-09:30:47.123" milliseconds TxnTm="20011217-09:30:47.123456" microseconds TxnTm="20011217-09:30:47.123456789" nanoseconds TxnTm="20011217-09:30:47.123456789123" picoseconds	

<table>
<tr><td>UTCTimeOnly

Base Type: String

XML Builtin: 0

XML Base: xs:time

XML Pattern[RS2]:</td><td>string field representing ~~T~~time-only ~~represented~~ in ~~UTC (~~Universal Time Coordinated (UTC), also known as Greenwich Mean Time ("GMT").
Its value space is described as the time of day in the Chapter 5.4 of ISO 8601.
Valid values are in the format ~~either HH:MM:SS (whole seconds) or~~ HH:MM:SS.s~~ss (milliseconds) format as specified in ISO 8601~~.

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

~~Valid values:~~

~~HH = 00-23, MM = 00-60 (60 only if UTC leap second), SS = 00-59. (without milliseconds)~~

where HH = 00-23 hours, MM = 00-59 minutes, SS = 00-60 seconds (60 only if UTC leap second), and optionally s (one or more digits representing a decimal fraction of a second)~~ss=000-999 (indicating milliseconds)~~.

The punctuation of “:” between hours minutes and seconds are required. The “.” is only required when sub-second time precision is specified.

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

Example(s)

~~MDEntry~~T~~i~~me="13:20:00~~.000~~" seconds
Tm="13:20:00.123"milliseconds
Tm="13:20:00.123456" microseconds
Tm="13:20:00.123456789" nanoseconds
Tm="13:20:00.123456789123" picoseconds</td><td>FIX.4.2</td></tr>
<tr><td>TZTimeOnly

Base Type: String

XML builtin: ~~0~~1

XML Base: xs:time

XML Pattern:</td><td>string field representing the time ~~represented~~ based on ISO 8601. This is the time with a Universal Time Coordinated(UTC) offset to allow identification of local time and timezone ~~of that time~~.
Its value space is described as the combination of date and time of day in the Chapter 5.4 of ISO 8601.
Valid values are in the f~~F~~ormat ~~is~~ HH:MM[:SS][Z | [+ | - hh[:mm]]] where HH = 00-23 hours, MM = 00-59 minutes, SS = 00-59 seconds, hh = 01-12 offset hours, mm = 00-59 offset minutes.</td><td>FIX.4.4 EP-1</td></tr>
</table>

	The punctuation of ":" are required. The "Z" or "+" or "-" are optional to denote a time zone offset. Example(s) MatTm="07:39Z" is 07:39 UTC MatTm="02:39-05" is five hours behind UTC, ~~thus~~ Eastern Time MatTm="15:39+08" is eight hours ahead of UTC, Hong Kong/Singapore time MatTm="13:09+05:30" is 5.5 hours ahead of UTC, India time	
TZTimestamp Base Type: String XML builtin: ~~0~~1 XML Base: xs:dateTime XML Pattern:	string field representing a ~~time/~~date and time combination ~~representing~~ in local time with an optional offset to Univeral Time Coordinated (UTC) ~~to allow identification of local time and timezone offset of that time. The representation~~ Its vaue space is described as the combination of date and time of day in the Chapter 5.4 of ~~based on~~ ISO 8601. Valid values are in the f~~F~~ormat ~~is~~ YYYY-MM-DD~~-~~THH:MM:SS.s~~*~~[Z | [+ | - hh[:mm]]] where YYYY = 0000 to 9999 year, MM = 01-12 month, DD = 01-31 day, HH = 00-23 hour~~s~~, MM = 00-59 minute~~s~~, SS = 00-59 seconds~~, hh = 01-12 offset hours, mm = 00-59 offset minutes~~, and optionally s~~ss~~ (one or more digits representing a decimal fraction of a second), hh = 01-12 offset hours, mm = 00-59 offset minutes. The punctuation of "-", ":" and the string value of "T" to separate the date and time are required. The "." is only required when sub-second time precision is specified. The "Z" or "+" or "-" are optional to denote an optional time zone offset. Example(s) "2006-09-01~~-~~T07:39Z" is 07:39 UTC on 1st of September 2006 "2006-09-01~~-~~T02:39-05" is five hours behind UTC, ~~thus~~ Eastern Time on 1st of September 2006 "2006-09-01~~-~~T15:39+08" is eight hours ahead of UTC, Hong Kong/Singapore time on 1st of September 2006 "2006-09-01~~-~~T13:09+05:30" is 5.5 hours ahead of UTC, India time on 1st of September 2006 Using decimal seconds: "2006-09-01T13:09.123+05:30" milliseconds "2006-09-01T13:09.123456+05:30" microseconds "2006-09-01T13:09.123456789+05:30" nanoseconds "2006-09-01T13:09.123456789123+05:30" picoseconds "2006-09-01T13:09.123456789Z" nanoseconds UTC timezone	FIX.4.4 EP-1

LocalMktDate Base Type: String XML builtin: 0 XML Base: xs:date XML Pattern:	string field representing a Date of Local Market (as opposed to UTC) in YYYY-MM-DD format. This is the "normal" date field used by the FIX Protocol. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31. Example(s) ~~BizDate="2003-09-10"~~ MaturityDate(541)="2015-07-24"	FIX.4.2

2.2 Existing FIX fields that require conveying timestamps with microseconds' granularity:

Summary:

Datatype	# of FIX fields
UTCTimestamp	42
UTCTimeOnly	9
TZTimestamp	1

2.2.1 Existing fields of data-type: UTCTimestamp

#	Added in FIX version	FIX tag	Field name	Datatype
1	FIX.2.7	42	OrigTime	UTCTimestamp
2	FIX.2.7	52	SendingTime	UTCTimestamp
3	FIX.2.7	60	TransactTime	UTCTimestamp
4	FIX.2.7	62	ValidUntilTime	UTCTimestamp
5	FIX.4.0	122	OrigSendingTime	UTCTimestamp
6	FIX.4.0	126	ExpireTime	UTCTimestamp
7	FIX.4.1	168	EffectiveTime	UTCTimestamp
8	FIX.4.2	341	TradSesStartTime	UTCTimestamp
9	FIX.4.2	342	TradSesOpenTime	UTCTimestamp
10	FIX.4.2	343	TradSesPreCloseTime	UTCTimestamp
11	FIX.4.2	344	TradSesCloseTime	UTCTimestamp
12	FIX.4.2	345	TradSesEndTime	UTCTimestamp
13	FIX.4.2	367	QuoteSetValidUntilTime	UTCTimestamp
14	FIX.4.2	438	ContraTradeTime	UTCTimestamp
15	FIX.4.2	443	StrikeTime	UTCTimestamp
16	FIX.4.3	483	TransBkdTime	UTCTimestamp
17	FIX.4.3	515	ExecValuationPoint	UTCTimestamp
18	FIX.4.3	586	OrigOrdModTime	UTCTimestamp
19	FIX.4.3	629	HopSendingTime	UTCTimestamp
20	FIX.4.4	769	TrdRegTimestamp	UTCTimestamp
21	FIX.4.4	779	LastUpdateTime	UTCTimestamp
22	FIX.4.4	962	SideTimeInForce	UTCTimestamp
23	FIX.4.4	1012	SideTrdRegTimestamp	UTCTimestamp
24	FIX.5.0	1145	EventTime	UTCTimestamp
25	FIX.5.0	1289	DerivativeEventTime	UTCTimestamp
~~26~~	~~FIX.5.0SP1~~	~~1492~~	~~ComplexEventStartDate~~	~~UTCTimestamp~~
27	FIX.5.0SP1	1493	ComplexEventEndDate	UTCTimestamp

28	FIX.5.0SP1	1504	RelSymTransactTime	UTCTimestamp
29	FIX.5.0SP2	1888	TradeMatchTimestamp	UTCTimestamp
30	FIX.5.0SP2	1914	ResponseTime	UTCTimestamp
31	FIX.5.0SP2	1915	QuoteDisplayTime	UTCTimestamp
32	FIX.5.0SP2	1984	UnderlyingEventTime	UTCTimestamp
~~33~~	~~FIX.5.0SP2~~	~~2054~~	~~UnderlyingComplexEventStartDate~~	~~UTCTimestamp~~
~~34~~	~~FIX.5.0SP2~~	~~2055~~	~~UnderlyingComplexEventEndDate~~	~~UTCTimestamp~~
35	FIX.5.0SP2	2062	LegEventTime	UTCTimestamp
36	FIX.5.0SP2	2116	NextAuctionTime	UTCTimestamp
37	FIX.5.0SP2	2251	LegComplexEventStartDate	UTCTimestamp
38	FIX.5.0SP2	2252	LegComplexEventEndDate	UTCTimestamp
39	FIX.5.0SP2	2445	AggressorTime	UTCTimestamp
40	FIX.5.0SP2	2468	MDStatisticStartDate	UTCTimestamp
41	FIX.5.0SP2	2469	MDStatisticEndDate	UTCTimestamp
42	FIX.5.0SP2	2476	MDStatisticTime	UTCTimestamp
43	FIX.4.2	273	MDEntryTime	UTCTimeOnly
44	FIX.5.0SP1	1495	ComplexEventStartTime	UTCTimeOnly
45	FIX.5.0SP1	1496	ComplexEventEndTime	UTCTimeOnly
46	FIX.5.0SP2	2057	UnderlyingComplexEventStartTime	UTCTimeOnly
47	FIX.5.0SP2	2058	UnderlyingComplexEventEndTime	UTCTimeOnly
48	FIX.5.0SP2	2204	LegComplexEventStartTime	UTCTimeOnly
49	FIX.5.0SP2	2247	LegComplexEventEndTime	UTCTimeOnly
50	FIX.5.0SP2	2470	MDStatisticStartTime	UTCTimeOnly
51	FIX.5.0SP2	2471	MDStatisticEndTime	UTCTimeOnly
52	FIX.4.4	1079	MaturityTime	TZTimeOnly
53	FIX.5.0	1213	UnderlyingMaturityTime	TZTimeOnly
54	FIX.5.0	1212	LegMaturityTime	TZTimeOnly
55	FIX.5.0	1253	DerivativeMaturityTime	TZTimeOnly
56	FIX.5.0	1405	UnderlyingLegMaturityTime	TZTimeOnly
57	FIX.5.0SP2	1550	InstrumentScopeMaturityTime	TZTimeOnly
58	FIX.4.4	1132	TZTransactTime	TZTimestamp

3 Issues and Discussion Points

3.1 Implementation alternatives

This section discusses two main alternatives to fulfil the business requirement:

1. Change existing fields' format. We considered two alternatives format:
   1.a. ISO 8061 with the following formats possibilities:
      I. 2015-10-13T17:53:03.123456789+00:00
      II. 2015-10-13T17:53:03.123456789Z
      III. 20151013T175303123456789Z

   1.b. 'FIX like' existing format e.g.: 20151013-17:53:03.123456789
   1.c. 'Meta type' encoding to allow legacy standard usage to remain compliant while enabling new formats.
   A new optional field "TimeFormat" is to be defined, default value 0 implies "FIX-like" with a value of 1 implying ISO formatting. Each Time or Date field that is not explicitly encoded otherwise, would become Type MetaXxxXxxx (e.g. MetaDateTime, MetaTimeOnly). The underlying format is then determined by the combination of TimeFormat and MetaType. Note that the representations do not change from those specified for options a and b, but the option of updating older standards to allow type a (ISO) dates is enabled without breaking the FIX conformant status of existing flows.
2. Maintain existing fields with no change, yet add new time fields to convey either:
   2.a. The entire date-time format in ISO 8061 format (duplicate information). e.g.
      I. 2015-10-13T17:53:03.123456789Z

or

   2.b. The fractions of seconds that are missing. e.g.
      I. 0.123456 (for microseconds)
      II. 0.123456789 (for nanoseconds)

The following table provide example how TransactTime(60) is represented in each of the above alternatives:

#		FIX Format example		FIX Format example		Metatype format example
1	a	|60=2015-10-13T17:53:03.123456789Z|	b	|60=20151013-17:53:03.123456789|	c	TBD=1|60=2015-10-13T17:53:03.123456789Z OR 60=20151013-17:53:03.123|[1]
2	a	|60=20151013-17:53:03123| TBD=2015-10-13T17:53:03.123456789Z|	b	|60=20151013-17:53:03123| TBD=0.123456789|		n/a

TBD – FIX tag to be defined

[1] Note that the optional field [TBD] is not used and is thus assumed as the default value of "FIX-like"

3.1.1 Implementation alternatives analysis

The following table summarise few of the aspects of the above implementation alternatives:

<table>
<tr><th rowspan="2">Feature</th><th colspan="3">1. Existing fields</th><th colspan="2">2. New fields</th></tr>
<tr><th>a. ISO 8061</th><th>b. FIX like</th><th>c. Metatype</th><th>a. Entire time</th><th>b. Fractions only</th></tr>
<tr><td>Example:</td><td>|60=2015-10-13T17:53:03.123456789Z|</td><td>|60=20151013-17:53:03.123456789|</td><td>TBD=1|60=2015-10-13T17:53:03.123456789Z
OR
TBD=0|60=20151013-17:53:03.123|</td><td>|60=20151013-17:53:03123|
TBD=2015-10-13T17:53:03.123456789Z|</td><td>|60=20151013-17:53:03123|
TBD=0.123456789|</td></tr>
<tr><td>Backwards compatibility</td><td colspan="3">FIX engines are non-backwards compatible! i.e. Existing FIX engine requires code changes

It is expected that a FIX engine that supports the new timestamps many not be able to communicate with a FIX engine that does not support the new format.
option a – all existing implementations would be broken.
option b – existing flows with 3 decimal places will continue to be valid.
option c – TBD field is optional for type 0 – default. Allowing existing implementations to remain FIX compliant while the standard evolves to support ISO standard.</td><td colspan="2">FIX engines backwards compatible
i.e. Existing FIX engine will pass the new fields to the application level.</td></tr>
<tr><td>Duplicate information</td><td colspan="3">No duplicated time fields</td><td colspan="2">Existing required time fields are duplicated on the wire.
In such a case consistency should be validated (i.e. the two fields refer to the same time in different format)</td></tr>
<tr><td>Wire format length</td><td>Short</td><td>Shortest</td><td>Short</td><td>Longest</td><td>Long</td></tr>
<tr><td>Other encodings</td><td colspan="3">Matches other encodings implementations
(SBE, ASN.1, GPB)</td><td colspan="2">Other encodings implementations do not require any additional fields</td></tr>
<tr><td>Date format</td><td>Should we amend dates data-types: UTCDateOnly and LocalMktDate?</td><td></td><td>Adds two new data types for each Date/Time field. A meta type and the new implementation type</td><td></td><td></td></tr>
</table>

3.1.2 Backwards compatibility analysis

We are aware that alternative #1 requires code changes for all existing FIX engine. Moreover, without such code modifications, exis~~i~~ting FIX engines will raise an exception and most likely terminate the session as soon as they receive a timestamps with more precisions than milliseconds.

In version FIX 5.0SP2 (FIXT1.2 or higher) we propose that FIX engine will implement and convey DefaultApplVerID(1137) and DefaultApplExtID(1407) at logon, and will verify that counterparty supports and extension pack with these enhancements (otherwise will terminate the session after a logout message). Older FIX version will not be able to implement such a validation.

3.2 Maximum Precision

Is there a need to establish an upper bound on the number of digits permitted in the decimal second fields to provide implemente~~o~~rs and users some level of surety regarding their implementations? It would seem that picoseconds would be a reasonable upper limit (12 digits of precision) for the foreseeable future and permit implemente~~o~~rs to structure their code accordingly. Allowing infinite precision would likely result in not having to update the specification, but implemente~~o~~rs and those creating persistence mechanisms at the datatype level will need to define some limit. It is likely that the first question to be asked by implemente~~o~~rs will be what is the maximum allowed number of decimal places.

3.2.1 Number Precision

Is there a need to support different number of precisions other than the most common use?

- Milliseconds – 3 precision
- Microseconds – 6 precision
- Nanoseconds – 9 precision
- Picoseconds – 12 precision

3.3 References external discussions:

3.3.1 Timestamp discussions:

http://www.fixtradingcommunity.org/pg/discussions/topicpost/168017/increasing-the-resolution-of-utc-timestamps-within-fix

http://www.fixtradingcommunity.org/pg/discussions/topicpost/2662115/microseconds

3.3.2 LocalMktDate discussions:

http://www.fixtradingcommunity.org/pg/discussions/topicpost/165028/localmktdate

http://www.fixtradingcommunity.org/pg/discussions/topicpost/165093/localmktdate-xsdate

3.4 Existing Implementation of higher precisions

The following table contains a list of venues that implemented their protocol with higher timestamp resolution.

Venue	Timestamp Resolution	Matches this proposal option	Reference
BATS US[2]	Microseconds	1.b	http://cdn.batstrading.com/resources/membership/BATS_US_Options_BZX_FIX_Specification.pdf and http://cdn.batstrading.com/resources/release_notes/2011/BATS-Introduces-Support-for-FIX-Microsecond-Timestamp-Granularity.pdf
CME	Nanoseconds	2.a	http://www.cmegroup.com/confluence/display/EPICSANDBOX/New+iLink+Architecture#NewiLinkArchitecture-iLinkMarketSegmentGatewayOverview
NASDAQ	Nanoseconds	1.b	http://www.nasdaqomx.com/digitalAssets/99/99437_fix---protocol-changes.pdf
NASDAQ	Nanoseconds	N/A	http://www.nasdaqomx.com/digitalAssets/100/100114_ouch-for-nasdaq-nordic-4.00.5.pdf

4 Recommended Approach

The author of this document, after considering all the above, recommends to proceed with 1.b option: 'FIX like' existing format e.g.: 20151013-17:53:03.123456789.
In considering ISO formatting, especially with regard to rewriting older FIX standards, we would draw attention to the risk of making existing and established FIX users non-compliant.

4.1 Time Precision Conservation Principle

The FIX Community should actively discourage increased time precision beyond the minimum required to meet business, technical, or regulatory requirements. Increase time precision creates a technical and operational burden in the areas of clock synchronization, increased message size, and increased storage for message logs.
Further, although this recommendation includes support for picosecond time resolution, implementing picosecond time resolution is not fully possible. Picosecond timing is supported for high frequency trading firms that must order events below nanosecond precision and for the recording of partial nanoseconds, typically encountered when constructing timestamps based on CPU counters where wall clock time is converted from CPU cycles.~~Picosecond timing is supported for high frequency trading firms that must order events below nanosecond precision. Likely these will be events that are created within a few clock cycles within a CPU, not events that cross messaging boundaries with counterparties.~~

[2] Clients may configure higher timestamp resolution

4.2 Maximum Time Precision

The enhanced time precision should be viewed as a maximum time precision during a FIX Session. Not all timestamp fields that use the UTCTimestamp and related datatypes require increased precision. Based upon the principal of conserving time precision to the minimum required, implementers are encouraged to only implement sufficient precision, up to the maximum agreed upon precision, during a FIX session. This means that the precision will vary across messages.

Rules of engagement may further limit specific FIX fields time precisions. So that although the service supports microseconds, some fields may be limited to milliseconds resolution, e.g. it is possible to limit ExpireTime(126) to milliseconds, whilst the TransactTime(60) is in microseconds.

4.3 Recommendations on decimal increments

Implementations are recommended to use one of the standard number of decimal place increments defined in the following table

Number of Decimal Places	Timestamp Resolution	Note
3	Milliseconds	Current FIX tag=value maximum
6	Microseconds	Required by MiFID regulations
9	Nanoseconds	In use at some venues now
12	Picoseconds	Usage discouraged due to operational and technical burden

4.4 Recommendations on trimming

Implementations shall not be required to trim trailing zeroes.

4.5 Recommendations on padding

Implementations shall not be required to pad to the session maximum precision. Implementations shall pad to the nearest millisecond, microsecond, nanosecond, picosecond boundary.

5 Proposed Message Flow

6 FIX Message Tables

7 FIX Component Blocks

8 Category Changes

None.

9 FIX Specification Errata

This section includes errata from prior versions and extension packs (EP) that are being implemented as corrections as part of this extension pack.

Jira Item	**Affected EP/Version**	**Synopsis of change.**
SPEC-2159	5.0 SP2	Correct the examples for the int data type and remove extraneous sentences from int, float and char descriptions.

Appendix A - Data Dictionary

Tag	FieldName	Action	Datatype	Description	FIXML Abbreviation	Add to / Deprecate from Message type or Component block

Appendix B - Glossary Entries

None.

Appendix C - Abbreviations

None.

Appendix D - Usage Examples

None.