



DPC: 24/30502290 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 18 October 2024

Origin: International

Latest date for receipt of comments: 08 December 2024

Project No. 2024/02194

Responsible committee: ICT/1 Information systems co-ordination

Interested committees:

Title: Draft BS ISO/IEC 25390 Information technology — Financial information exchange — Simple binary encoding.

Please notify admin.start@bsigroup.com if you are aware of any keywords that might assist in clarifying or identifying the standard or of the content of this standard:

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 08 December 2024

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

E-mail: admin.start@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/ Figure/Table/Note	Type of comment	Comment (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



DRAFT International Standard

Information technology — Financial information exchange — Simple binary encoding

ICS: 35.240.40

ISO/IEC DIS 25390

ISO/IEC JTC 1

Secretariat: **ANSI**

Voting begins on:
2024-10-15

Voting terminates on:
2025-01-07

This document is circulated as received from the committee secretariat.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

Reference number
ISO/IEC DIS 25390:2024(en)

Full document details: 24/30502290 DC

© ISO/IEC 2024  **Documents**
Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Table of Contents

Table of Contents..... 2

Foreword 7

Introduction 8

1 Scope..... 9

2 Normative references 9

3 Terms and definitions..... 9

 3.1 datatype 9

 3.2 encoding..... 9

 3.3 field..... 9

 3.4 message schema..... 9

 3.5 message template..... 9

 3.6 session protocol 10

 3.7 XML schema 10

 3.8 Specification terms 10

4 Objectives..... 10

 4.1 General..... 10

 4.2 Binary type system 10

 4.3 Design principles 10

 4.4 Message schema..... 11

 4.5 Documentation 11

 4.5.1 General..... 11

 4.5.2 Document format..... 11

5 Field Encoding..... 11

 5.1 Field aspects..... 11

 5.1.1 General..... 11

 5.1.2 Semantic datatype 11

 5.1.3 Encoding..... 12

 5.1.4 Metadata..... 12

 5.1.5 Field presence..... 12

 5.1.6 Default value 12

 5.2 FIX datatype summary..... 12

 5.3 Common field schema attributes..... 15

 5.3.1 General..... 15

 5.3.2 Inherited attributes 16

 5.3.3 Non-FIX types 16

 5.4 Integer encoding..... 16

 5.4.1 General..... 16

 5.4.2 Primitive type encodings 16

 5.4.3 Range attributes for integer fields..... 16

 5.4.4 Byte order..... 17

 5.4.5 Integer encoding specifications 17

 5.4.6 Examples of integer fields..... 17

 5.5 Decimal encoding 18

 5.5.1 General..... 18

 5.5.2 Composite encodings 18

 5.5.3 Range attributes for decimal fields 18

 5.5.4 Encoding specifications for decimal types 19

 5.5.5 Composite encoding padding..... 19

 5.5.6 Examples of decimal fields..... 19

 5.6 Float encoding..... 20

 5.6.1 General..... 20

 5.6.2 Primitive types..... 20

5.6.3	Null values.....	20
5.6.4	Byte order.....	20
5.6.5	Float encoding specifications.....	20
5.6.6	Examples of floating point fields.....	20
5.7	String encodings.....	20
5.7.1	General.....	20
5.7.2	Character.....	21
5.7.3	Fixed-length character array.....	21
5.7.4	Variable-length string encoding.....	22
5.7.5	Range attributes for string Length.....	22
5.7.6	Encoding specifications for variable-length string.....	22
5.7.7	Example of a variable-length string field.....	23
5.8	Data encodings.....	23
5.8.1	General.....	23
5.8.2	Fixed-length data.....	23
5.8.3	Variable-length data encoding.....	23
5.8.4	Range attributes for variable-length data Length.....	24
5.8.5	Encoding specifications for variable-length data.....	24
5.8.6	Example of a data field.....	24
5.9	MonthYear encoding.....	24
5.9.1	General.....	24
5.9.2	Composite encoding padding.....	25
5.9.3	Encoding specifications for MonthYear.....	25
5.10	Date and time encoding.....	25
5.10.1	General.....	25
5.10.2	Epoch.....	25
5.10.3	Time unit.....	25
5.10.4	Encoding specifications for date and time.....	26
5.10.5	Examples of date/time fields.....	27
5.11	Local date encoding.....	27
5.12	Local time encoding.....	27
5.12.1	General.....	27
5.12.2	TZTimestamp encoding.....	28
5.12.3	Composite encoding padding.....	28
5.12.4	TZTimeOnly encoding.....	28
5.12.5	Composite encoding padding.....	29
5.13	Enumeration encoding.....	29
5.13.1	General.....	29
5.13.2	Primitive type encodings.....	29
5.13.3	Value encoding.....	29
5.13.4	Encoding specification of enumeration.....	29
5.13.5	Enumeration examples.....	30
5.13.6	Constant field of an enumeration value.....	30
5.13.7	Boolean encoding.....	30
5.14	Multi-value choice encoding.....	31
5.14.1	General.....	31
5.14.2	Primitive type encodings.....	31
5.14.3	Value encoding.....	31
5.14.4	Encoding specification of multi-value choice.....	31
5.14.5	Multi-value example.....	31
5.15	Field value validation.....	32
6	Message Structure	32
6.1	Message Framing.....	32
6.1.1	General.....	32
6.1.2	Simple Open Framing Header.....	33
6.2	SBE Message Encoding Header.....	33
6.2.1	General.....	33
6.2.2	Message header schema.....	34

6.2.3	Root block length	34
6.2.4	Template ID	34
6.2.5	Schema ID	34
6.2.6	Schema version	34
6.2.7	Number of repeating groups	35
6.2.8	Number of variable-length fields	35
6.3	Message Body	35
6.3.1	General	35
6.3.2	Data only on the wire	35
6.3.3	Direct access	35
6.3.4	Field position and padding	35
6.4	Repeating Groups	36
6.4.1	General	36
6.4.2	Schema specification of a group	36
6.4.3	Group block length	36
6.4.4	Padding at end of a group entry	37
6.4.5	Entry counter	37
6.4.6	Empty group	37
6.4.7	Multiple repeating groups	37
6.4.8	Nested repeating group specification	37
6.4.9	Nested repeating group wire format	37
6.4.10	Empty group means nested group is empty	38
6.4.11	Group dimension encoding	38
6.5	Sequence of message body elements	39
6.5.1	Root level elements	39
6.5.2	Repeating group elements	39
6.6	Message structure validation	39
7	Message Schema	40
7.1	XML schema for SBE message schemas	40
7.2	XML namespace	40
7.3	Naming convention	40
7.3.1	General	40
7.3.2	Capitalization	40
7.4	Root element	40
7.4.1	General	40
7.4.2	<messageSchema> attributes	40
7.4.3	Schema versioning	41
7.5	Data encodings	41
7.5.1	Encoding sets	41
7.5.2	Encoding name	41
7.5.3	Simple encodings	42
7.5.4	General	42
7.5.5	Composite encodings	43
7.5.6	Reference to reusable types	44
7.5.7	Enumeration encodings	45
7.5.8	Multi-value choice encodings (bitset)	47
7.6	Message template	48
7.6.1	General	48
7.6.2	Reserved space	48
7.6.3	Message members	48
7.6.4	Member order	49
7.6.5	<message> element attributes	49
7.7	Field attributes	49
7.8	Repeating group schema	51
7.9	Schema validation	51
7.9.1	General	51
7.9.2	Message with a repeating group	52
7.9.3	Message with raw data fields	52

7.10	Reserved element names.....	53
7.10.1	Composite types.....	53
7.10.2	Composite type elements.....	53
8	Schema Extension Mechanism.....	53
8.1	Objective.....	53
8.1.1	General.....	53
8.1.2	Constraints.....	54
8.2	Message schema features for extension.....	54
8.2.1	Schema version.....	54
8.2.2	Since version.....	54
8.2.3	Block length.....	54
8.2.4	Deprecated elements.....	55
8.3	Wire format features for extension.....	55
8.3.1	Block size.....	55
8.3.2	Number of repeating groups and variable data.....	55
8.4	Compatibility strategy.....	55
8.5	Message schema extension example.....	55
8.5.1	Initial version of a message schema.....	55
8.5.2	Second version - a new message is added.....	56
8.5.3	Third version - a field is added.....	56
9	Usage Guidelines.....	56
9.1	Identifier encodings.....	56
10	Examples.....	57
10.1	General.....	57
10.2	Flat, fixed-length message.....	57
10.2.1	General.....	57
10.2.2	Sample order message schema.....	57
10.2.3	Wire format of an order message.....	58
10.3	Message with a repeating group.....	59
10.3.1	General.....	59
10.3.2	Sample execution report message schema.....	59
10.3.3	Wire format of an execution message.....	61
10.3.4	Interpretation.....	61
10.4	Message with a variable-length field.....	62
10.4.1	Sample business reject message schema.....	62
10.4.2	Wire format of a business reject message.....	62
10.4.3	Interpretation.....	62
	Bibliography.....	64

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

IEC Secretariat

3, rue de Varembé
CH-1211 Geneva 20
Phone: +41 22 919 02 11
Fax: +41 22 919 03 00
Email: info@iec.ch
Website: www.iec.ch

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the Joint Development Foundation (as OpenChain Specification) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The Financial Information eXchange Simple Binary Encoding (SBE) targets high performance trading systems. It is optimized for low latency of encoding and decoding while keeping bandwidth utilization reasonably small. For compatibility, it is intended to represent all FIX semantics. SBE is primarily a fixed length wire format but also supports variable length fields and repeating groups with fixed length entries. The wire format does not contain any meta-data other than length information for variable elements. The meta-data is a message schema provided out-of-band as an XML Schema Definition (XSD) file.

Financial Services – Financial Information eXchange – Simple Binary Encoding

1 Scope

This document provides the normative specification of Simple Binary Encoding (SBE), which is one of the possible syntaxes for FIX messages, but not limited to FIX messages. The scope comprises the encoding (wire format) and the message schema for SBE.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

— IETF RFC 2119 – *Key words for use in RFCs to Indicate Requirement Levels* March 1997

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 11404 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at <https://www.iso.org/obp>

— IEC Electropedia: available at <http://www.electropedia.org/>

3.1 datatype

field type with its associated encoding attributes

Note 1 to entry: Includes backing primitive types and valid values or range. Some types have additional attributes, e.g. epoch of a date.

3.2 encoding

message format for interchange

Note 1 to entry: The term is commonly used to mean the conversion of one data format to another, such as text to binary. However, SBE strives to use native binary datatypes in order to make conversion unnecessary, or at least trivial.

Note 2 to entry: Encoding also refers to the act of formatting a message, as opposed to decoding.

3.3 field

unit of data contained by a message

3.4 message schema

metadata that specifies messages and their data types and identifiers

Note 1 to entry: Message schemas may be disseminated out of band.

Note 2 to entry: For SBE, message schemas are expressed as an XML document that conforms to an XML schema that is published as part of this standard.

3.5 message template

metadata that specifies the fields that belong to one particular message type

Note 1 to entry: A message template is contained by a message schema.

3.6 session protocol

protocol concerned with the reliable delivery of messages over a transport.

Note 1 to entry: FIX makes a distinction between session protocol and the encoding of a message payload, as described by this document. See the [specifications section](#) of the FIX Protocol web site for supported session protocols and encodings.

3.7 XML schema

defines the elements and attributes that may appear in an XML document.

Note 1 to entry: The SBE message schema is defined in W3C (XSD) schema language since it is the most widely adopted format for XML schemas.

3.8 Specification terms

These key words in this document are to be interpreted as described in IETF RFC 2119.

4 Objectives

4.1 General

SBE was designed to meet the needs of high performance trading systems, but it may also be applicable to other applications with similar performance characteristics. It is optimized for low latency of encoding and decoding while keeping bandwidth utilization reasonably small. For compatibility, it is intended to represent all FIX semantics.

This encoding specification describes the wire protocol for messages. Thus, it provides a standard for interoperability between communicating parties. Users are free to implement the standard in a way that best suits their needs.

The encoding standard is complementary to other FIX standards for session protocol and application level behavior.

4.2 Binary type system

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

The binary type system has been enhanced in these ways:

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

4.3 Design principles

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

- Usage of native binary datatypes and simple types derived from native binaries, such as prices and timestamps.

- Preference for fixed positions and fixed length fields, supporting direct access to data and avoiding the need for management of heaps of variable-length elements which must be sequentially processed.

4.4 Message schema

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

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

4.5 Documentation

4.5.1 General

This document explains:

- The binary type system for field encoding
- Message structure, including field arrangement, repeating groups, and relationship to a message header that may be provided by a session protocol.
- The SBE message schema.

4.5.2 Document format

In this document, these formats are used for technical specifications and data examples.

This is a sample encoding specification

```
<type name="short" primitiveType="int16" semanticType="int"/>
```

This is sample data as it would be transmitted on the wire

```
10270000
```

5 Field Encoding

5.1 Field aspects

5.1.1 General

Every field has the following aspects: semantic datatype, encoding, and metadata. They will be specified in more detail in the sections on datatype encoding and message schema but are introduced here as an overview.

5.1.2 Semantic datatype

5.1.2.1 FIX datatype

The FIX semantic datatype of a field tells a data domain in a broad sense, for example, whether it is numeric or character data, or whether it represents a time or price. Simple Binary Encoding represents all of the semantic datatypes that FIX protocol has defined across all encodings. In message specifications, FIX datatype is declared with attribute semanticType. See the section [FIX datatype summary](#) for details.

5.1.2.2 Generic datatype

A datatype is defined as a combination of a value space and a lexical space. Value space is the range of its possible values while lexical space is how those values are represented in a message encoding, in this case SBE. Value space of datatypes is defined using the vocabulary in ISO/IEC 11404:2007 *Information technology – General-Purpose Datatypes (GPD)*. That standard defines types independently of platform and programming language.

5.1.3 Encoding

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

5.1.4 Metadata

Field metadata, part of a message schema, describes a field to application developers. Elements of field metadata are:

- Field ID, also known as FIX tag number, is a unique identifier of a field for semantic purposes. For example, tag 55 identifies the Symbol field of an instrument.
- Field name, as it is known in FIX specifications
- The FIX semantic datatype and encoding type that it maps to
- Valid values or data range accepted
- Documentation

Metadata is normally *not* sent on the wire with Simple Binary Encoding messages. It is necessary to possess the message schema that was used to encode a message in order to decode it. In other words, Simple Binary Encoding messages are not self-describing. Rather, message schemas are typically exchanged out-of-band between counterparties.

See section [Message Schema](#) for details.

5.1.5 Field presence

By default, fields are assumed to be required in a message. However, fields may be specified as optional. To indicate that a value is not set, a special null indicator value is sent on the wire. The null value varies according to datatype and encoding. Global defaults for null value may be overridden in a message schema by explicitly specifying the value that indicates nullness.

Alternatively, fields may be specified as constant. In which case, the data is not sent on the wire, but may be treated as constants by applications.

5.1.6 Default value

Default value handling is not specified by the encoding layer. A null value of an optional field does not necessarily imply that a default value should be applied. Rather, default handling is left to application layer specifications.

5.2 FIX datatype summary

FIX semantic types are mapped to binary field encodings as follows. See sections below for more detail about each type.

Schema attributes may restrict the range of valid values for a field. See section [Common field schema attributes](#) for details.

FIX semantic type	Binary type	Value space (ISO/IEC 11404:2007)	Description
int	Integer encoding	integer range	An integer number
Length	Integer encoding	size	Field length in octets. Value must be non-negative.

FIX semantic type	Binary type	Value space (ISO/IEC 11404:2007)	Description
TagNum	Integer encoding	ordinal	A field's tag number. Value must be positive.
SeqNum	Integer encoding	ordinal	A field representing a message sequence number. Value must be positive
NumInGroup	Group dimension encoding	size	A counter representing the number of entries in a repeating group. Value must be positive.
DayOfMonth	Integer encoding	integer range 1..31	A field representing a day during a particular month (values 1 to 31).
Qty	Decimal encoding	Scaled	A number representing quantity of a security, such as shares. The encoding may constrain values to integers, if desired.
float	Float encoding	real	A real number with binary representation of specified precision
Price	Decimal encoding	Scaled	A decimal number representing a price
PriceOffset	Decimal encoding	Scaled	A decimal number representing a price offset, which can be mathematically added to a Price.
Amt	Decimal encoding	Scaled	A field typically representing a Price times a Qty.
Percentage	Decimal encoding	Scaled	A field representing a percentage (e.g. 0.05 represents 5% and 0.9525 represents 95.25%).
char	Character	character	Single US-ASCII character value. Can include any alphanumeric character or punctuation. All char fields are case sensitive (i.e. m != M).
String	Fixed-length character array	array element = character	A fixed-length character array of ASCII encoding

FIX semantic type	Binary type	Value space (ISO/IEC 11404:2007)	Description
String	Variable-length data encoding	characterstring	Alpha-numeric free format strings can include any character or punctuation. All String fields are case sensitive (i.e. morstatt != Morstatt). ASCII encoding.
String— EncodedText	String encoding	characterstring	Non-ASCII string. The character encoding may be specified by a schema attribute.
XMLData	String encoding	characterstring	Variable-length XML. Must be paired with a Length field.
data	Fixed-length data	array element=octet	Fixed-length non-character data
data	Variable-length data encoding	octetstring	Variable-length data. Must be paired with a Length field.
Country	Fixed-length character array ; size = 2 or a subset of values may use Enumeration encoding	array (1..2) of character	ISO 3166-1:2013 Country code
Currency	Fixed-length character array ; size = 3 or a subset of values may use Enumeration encoding	array (1..3) of character	ISO 4217:2015 Currency code (3 character)
Exchange	Fixed-length character array ; size = 4 or a subset of values may use Enumeration encoding	array (1..4) of character	ISO 10383:2012 Market Identifier Code (MIC)
Language	Fixed-length character array ; size = 2 or a subset of values may use Enumeration encoding	array (1..2) of character	National language - uses ISO 639-1:2002 standard
Implicit enumeration— char or int	Enumeration encoding	state	A single choice of alternative values
Boolean	Boolean encoding	boolean	Values true or false
MultipleCharValue	Multi-value choice encoding	set element = character	Multiple choice of a set of values
MultipleStringValue	Multi-value choice encoding . String choices must be mapped to int values.	set element = characterstring	Multiple choice of a set of values
MonthYear	MonthYear encoding	record	A flexible date format that must include month and

FIX semantic type	Binary type	Value space (ISO/IEC 11404:2007)	Description
			year at least but may also include day or week.
UTCTimestamp	Date and time encoding	time	Time/date combination represented in UTC (Universal Time Coordinated, also known as "GMT")
UTCTimeOnly	Date and time encoding	time	Time-only represented in UTC (Universal Time Coordinated, also known as "GMT")
UTCDateOnly	Date and time encoding	time	Date represented in UTC (Universal Time Coordinated, also known as "GMT")
LocalMktDate	Local date encoding	time	Local date (as opposed to UTC)
TZTimeOnly	TZTimeOnly encoding	time	Time of day
TZTimestamp	TZTimestamp encoding	time	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 8601:2004

The FIX semantic types listed above are spelled and capitalized exactly as they are in the FIX repository from which official FIX documents and references are derived.

5.3 Common field schema attributes

5.3.1 General

Schema attributes alter the range of valid values for a field. Attributes are optional unless specified otherwise.

Schema attribute	Description
presence=required	The field must always be set. This is the default presence. Mutually exclusive with nullValue.
presence=constant	The field has a constant value that need not be transmitted on the wire. Mutually exclusive with value attributes.
presence=optional	The field need not be populated. A special null value indicates that a field is not set. The presence attribute may be specified on either on a field or its encoding.

Schema attribute	Description
nullValue	A special value that indicates that an optional value is not set. See encodings below for default nullValue for each type. Mutually exclusive with presence=required and constant.
minValue	The lowest valid value of a range. Applies to scalar datatypes, but not to String or datatypes.
maxValue	The highest valid value of a range (inclusive unless specified otherwise). Applies to scalar datatypes, but not to String or datatypes.
semanticType	Specifies the FIX semantic type of a field or encoding. It may be specified on either a field or its encoding.

5.3.2 Inherited attributes

The attributes listed above apply to a field element or its encoding (wire format). Any attributes specified on an encoding are inherited by fields that use that encoding.

5.3.3 Non-FIX types

Encodings may be added to SBE messages that do not correspond to listed FIX datatypes. In that case, the encoding and fields that use the encoding will not have a semanticType attribute.

5.4 Integer encoding

5.4.1 General

Integer encodings should be used for cardinal or ordinal number fields. Signed integers are encoded in a two's complement binary format.

5.4.2 Primitive type encodings

Numeric datatypes may be specified by range and signed or unsigned attribute. Integer types are intended to convey common platform primitive datatypes as they reside in memory. An integer type should be selected to hold the maximum range of values that a field is expected to hold.

Primitive type	Description	Length (octets)
int8	Signed byte	1
uint8	Unsigned byte / single-byte character	1
int16	16-bit signed integer	2
uint16	16-bit unsigned integer	2
int32	32-bit signed integer	4
uint32	32-bit unsigned integer	4
int64	64-bit signed integer	8
uint64	64-bit unsigned integer	8

5.4.3 Range attributes for integer fields

The default data range and null indicator are listed below for each integer encoding.

A message schema may optionally specify a more restricted range of valid values for a field.

For optional fields, a special null value is used to indicate that a field value is not set. The default null indicator may also be overridden by a message schema.

Required and optional fields of the same primitive type have the same data range. The null value must not be set for a required field.

Schema attribute	int8	uint8	int16	uint16	int32	uint32	int64	uint64
minValue	-127	0	-32767	0	$-2^{31} + 1$	0	$-2^{63} + 1$	0
maxValue	127	254	32767	65534	$2^{31} - 1$	$2^{32} - 2$	$2^{63} - 1$	$2^{64} - 2$
nullValue	-128	255	-32768	65535	-2^{31}	$2^{32} - 1$	-2^{63}	$2^{64} - 1$

5.4.4 Byte order

The byte order of integer fields, and for derived types that use integer components, is specified globally in a message schema. Little-Endian order is the default encoding, meaning that the least significant byte is serialized first on the wire.

See section [Message schema attributes](#) for the specification, including `byteOrder`. Message schema designers should specify the byte order most appropriate to their system architecture and that of their counterparties.

5.4.5 Integer encoding specifications

By nature, integers map to simple encodings. These are valid encoding specifications for each of the integer primitive types.

```
<type name="int8" primitiveType="int8" />
<type name="int16" primitiveType="int16" />
<type name="int32" primitiveType="int32" />
<type name="int64" primitiveType="int64" />
<type name="uint8" primitiveType="uint8" />
<type name="uint16" primitiveType="uint16" />
<type name="uint32" primitiveType="uint32" />
<type name="uint64" primitiveType="uint64" />
```

5.4.6 Examples of integer fields

Examples show example schemas and encoded bytes on the wire as hexadecimal digits in Little-Endian byte order.

Example integer field specification

```
<field type="uint32" name="ListSeqNo" id="67" semanticType="int" description="Order number within the list" />
```

Value on the wire - uint32 value decimal 10,000, hexadecimal 2710.

10270000

Optional field with a valid range 0-6

```
<type name="range06" primitiveType="uint8" maxValue="6" presence="optional" nullValue="255" />
<field type="range06" name="MaxPriceLevels" id="1090" semanticType="int"/>
```

Wire format of uint8 value decimal 3.

03

Sequence number field with integer encoding

```
<field type="uint64" name="MsgSeqNum" id="34" semanticType="SeqNum" />
```

Wire format of uint64 value decimal 100,000,000,000, hexadecimal 174876E800.

00e8764817000000

Wire format of uint16 value decimal 10000, hexadecimal 2710.

1027

Wire format of uint32 null value $2^{32} - 1$

ffffffff

5.5 Decimal encoding

5.5.1 General

Decimal encodings should be used for prices and related monetary data types like PriceOffset and Amt.

FIX specifies Qty as a float type to support fractional quantities. However, decimal encoding may be constrained to integer values if that is appropriate to the application or market.

5.5.2 Composite encodings

5.5.2.1 General

Prices are encoded as a scaled decimal, consisting of a signed integer mantissa and signed exponent. For example, a mantissa of 123456 and exponent of -4 represents the decimal number 12.3456.

5.5.2.2 Mantissa

Mantissa represents the significant digits of a decimal number. Mantissa is a commonly used term in computing, but it is properly known in mathematics as significand or coefficient.

5.5.2.3 Exponent

Exponent represents scale of a decimal number as a power of 10.

5.5.2.4 Floating point and fixed point encodings

A floating-point decimal transmits the exponent on the wire while a fixed-point decimal specifies a fixed exponent in a message schema. A constant negative exponent specifies a number of assumed decimal places to the right of the decimal point.

Implementations should support both 32-bit and 64-bit mantissa. The usage depends on the data range that must be represented for a particular application. It is expected that an 8-bit exponent should be sufficient for all FIX uses.

Encoding type	Description	Backing primitives	Length (octets)
decimal	Floating-point decimal	Composite: int64 mantissa, int8 exponent	9
decimal64	Fixed-point decimal	int64 mantissa, constant exponent	8
decimal32	Fixed-point decimal	int32 mantissa, constant exponent	4

Optionally, implementations may support any other signed integer types for mantissa and exponent.

5.5.3 Range attributes for decimal fields

The default data ranges and null indicator are listed below for each decimal encoding.

A message schema may optionally specify a more restricted range of valid values for a field. For optional fields, a special mantissa value is used to indicate that a field value is null.

Schema attribute	decimal	decimal64	decimal32
exponent range	-128 to 127	-128 to 127	-128 to 127
mantissa range	$-2^{63} + 1$ to $2^{63} - 1$	$-2^{63} + 1$ to $2^{63} - 1$	$-2^{31} + 1$ to $2^{31} - 1$