INTERNATIONAL STANDARD

ISO 3531-1

First edition
2022-04

Financial services — Financial information eXchange session layer —

Part 1: FIX tagvalue encoding

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

Reference number
ISO 3531-1:2022(E)

© ISO 2022

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

COPYRIGHT PROTECTED DOCUMENT

© ISO 2022

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

© ISO 2022 – All rights reserved

Contents

Page

Foreword ... **iv**
Introduction ... **v**
1 Scope ... **1**
2 Normative references ... **1**
3 Terms and definitions ... **1**
4 FIX tagvalue message syntax ... **2**
4.1 Character encoding ... 2
4.2 Field syntax ... 2
4.2.1 Tag (field identifier) ... 2
4.2.2 Tag delimiter ... 2
4.2.3 Field value ... 2
4.2.4 Field delimiter ... 2
4.2.5 Well-formed field ... 2
4.2.6 Example of a FIX tagvalue message ... 3
4.3 Message structure ... 3
4.3.1 General ... 3
4.3.2 Message type ... 3
4.3.3 Field presence ... 3
4.3.4 Field sequence ... 3
4.3.5 Message delimiter ... 4
4.3.6 Components ... 4
4.3.7 Groups and repeating groups ... 4
4.3.8 Encoded data fields ... 6
5 Standard header and trailer ... **7**
5.1 General ... 7
5.2 Standard header ... 7
5.2.1 General ... 7
5.2.2 Body length calculation ... 7
5.2.3 Standard header definition ... 7
5.3 Standard trailer ... 8
5.3.1 Standard trailer definition ... 8
5.3.2 Checksum ... 8
6 FIX tagvalue datatypes ... **8**
6.1 General ... 8
6.2 Value space ... 8
6.3 Lexical space ... 8
6.3.1 General ... 8
6.3.2 Character encoding ... 9
6.3.3 Lexical encoding for FIX datatypes ... 9
6.3.4 XML data ... 15
7 Code sets ... **15**
7.1 General ... 15
7.2 Underlying value type ... 15
7.2.1 General ... 15
7.2.2 Internal code sets ... 15
7.2.3 External code sets ... 15
Annex A (informative) **Checksum calculation** ... **16**
Bibliography ... **17**

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 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).

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

For an explanation of 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 FIX Trading Community (as FIX Session Layer Technical Specification) and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 68, *Financial services*, Subcommittee SC 9, *Information exchange for financial services*, and adopted under the "fast-track procedure".

A list of all parts in the ISO 3531 series can be found on the ISO website.

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.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Introduction

FIX session protocol was written to be independent of any specific communications protocol (e.g. X.25, async, TCP/IP) or physical medium (e.g. copper, fibre, satellite) chosen for electronic data delivery. It offers a reliable stream where a message is delivered once and in order. The FIX session layer is designed to survive and resume operation in the event of the loss of transport level connections caused by any type of failure, including network outage, application failure or computer hardware failures.

The session layer is concerned with the ordered delivery of data while the application level defines business-related data content. This document focuses on the ordered delivery of data using the "FIX session protocol".

The FIX session protocol is implemented using the FIX tagvalue encoding syntax for the standard header, standard trailer and the session level messages which make up the FIX session protocol. It is possible to send messages encoded using other FIX-defined encodings (e.g. FIXML, SBE, JSON, GPB, ASN.1) or other non-FIX-defined encodings (e.g. XML, FpML, ISO 20022 XML, JSON).

The Financial Information eXchange tagvalue encoding is the original encoding used for FIX messages. The tagvalue encoding is the encoding used by the FIX session layer; it corresponds to the Presentation Layer of the ISO Open Systems Interconnection model. The encoding uses an integer number known as a *tag* to identify the field, followed by the "=" character (hexadecimal 0x3D), then the value of that field encoded in the ISO/IEC 8859-1 character set. Each tagvalue pair is separated by the *Start of Heading* control character <SOH> (hexadecimal value 0x01), which is defined by ISO/IEC 6429. The tagvalue encoding also supports the encoding of binary and multibyte character data in certain encoded data fields that are preceded by a Length field.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

Financial services — Financial information eXchange session layer —

Part 1: FIX tagvalue encoding

1 Scope

This document provides the normative specification of the FIX tagvalue encoding, which is one of the possible syntaxes for FIX messages.

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.

ISO/IEC 11404, *Information technology — General-Purpose Datatypes (GPD)*

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
field presence
existence or use of a field within a message

Note 1 to entry: FIX specifications and rules of engagement based on FIX should refer to a field as being present in a message.

3.2
component presence
existence or use of a component within a message

Note 1 to entry: FIX specifications and rules of engagement based on FIX should refer to a component as being present in a message.

3.3
repeating group instance
specific record of the group within a repeating group

Note 1 to entry: Records are defined in ISO/IEC 11404.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

3.4
character digit
character representation of a number, 0 to 9, in the character set used for encoding

Note 1 to entry: Characters 0x30 to 0x39 in the Latin alphabet No. 1 character set (ISO/IEC 8859-1).

4 FIX tagvalue message syntax

4.1 Character encoding

With the exception of datatype *data*, tagvalue encoding uses a single-byte character set. By default, the encoding is ISO/IEC 8859-1, Latin alphabet No. 1.

By counterparty agreement, a different single-byte character set may be used.

Note that the Latin-1 alphabet is an 8-bit code but reserves two ranges for control codes. Message structure is supplemented by ISO/IEC 6429 control character set C0.

4.2 Field syntax

4.2.1 Tag (field identifier)

Each field is uniquely identified by an integer, known as a tag. Tags must be unique among both session and application message fields. (Fields in the standard header and standard trailer components are shared by session and application messages.)

Tags are serialized according to the syntax of the *TagNum* datatype.

4.2.2 Tag delimiter

A tag is delimited from its field value by the equals sign (=), character value 61 (decimal).

4.2.3 Field value

Field values are serialized according to their FIX datatype syntax.

4.2.4 Field delimiter

All fields in a FIX message, including those of datatype *data*, must be terminated by a delimiter character. The *Start of Heading* control character, value 0x01, referred to in this document as <SOH>, is used for field termination.

There must be no embedded <SOH> characters within field values except for those of datatype *data*.

4.2.5 Well-formed field

A well-formed field has the form:

tag=*value*<SOH>

A field shall be considered malformed if any of the following occurs as a result of encoding:

- the tag is empty;
- the tag delimiter is missing;
- the value is empty;
- the value contains an <SOH> character and the datatype of the field is not *data* or *XMLdata*;

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

- the datatype of the field is *data* and the field is not immediately preceded by its associated Length field.

4.2.6 Example of a FIX tagvalue message

The following is a FIX 4.2 NewOrderSingle(35=D) message in classic tagvalue pair format:

```
8=FIX.4.2<SOH>9=251<SOH>35=D<SOH>49=AFUNDMGR<SOH>56=ABROKER<SOH>
34=2<SOH>52=2003061501:14:49<SOH>11=12345<SOH>1=111111<SOH>63=0<SOH>
64=20030621<SOH>21=3<SOH>110=1000<SOH>111=50000<SOH>55=IBM<SOH>
48=459200101<SOH>22=1<SOH>54=1<SOH>60=2003061501:14:49<SOH>38=5000<SOH>
40=1<SOH>44=15.75<SOH>15=USD<SOH>59=0<SOH>10=127<SOH>
```

4.3 Message structure

4.3.1 General

A FIX message is a collection of fields that begins with the BeginString(8) field, followed by the BodyLength(9) field, then the MsgType(35) field, and ends with the Checksum(10) field. The message is identified by the value provided in the MsgType(35) field.

This clause summarizes general specifications for constructing messages in tagvalue syntax.

The general format of a message is a standard header followed by the message body fields and terminated with a standard trailer.

Each message is constructed of a stream of *tag=value* fields with a field delimiter between fields in the stream. Messages will be referenced as *message_name*(35=*x*) with *x* representing the message type; fields will be referenced as *field_name*(*tag*).

4.3.2 Message type

The MsgType(35) field is used to identify the type of message encoded. The definition and scope of the message type is provided by the encoder. For example, the FIX session layer standard defines a set of messages to initiate and manage a FIX session. The FIX application layer standard (commonly referred to as *FIX Latest*) defines additional message types for business level processing. There are no message types or reserved values for message types defined at the encoding level.

4.3.3 Field presence

In a message definition, a field must be specified as either required, optional or conditionally required. If it is conditionally required, the message specification must give a clear rule for when the field must be present.

All fields present in an encoded message must have a value. Optional fields without values must be omitted from the FIX message.

A tag (field) must appear at most once in a message, except when the tag appears within a repeating group.

A tag (field) must appear at most once per repeating group instance.

4.3.4 Field sequence

Except where noted, fields within a message can be defined in any sequence. (Relative position of a field within a message is inconsequential.) The exceptions to this rule are:

- General message format is composed of the standard header, followed by the body, followed by the standard trailer.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

- The first three fields in the StandardHeader component must be BeginString(8), followed by BodyLength(9), followed by MsgType(35), in that sequence.
- The last field in the standard trailer must be CheckSum(10).
- Within a repeating group, field sequence is strictly defined by a group definition.

4.3.5 Message delimiter

Messages are effectively delimited by the <SOH> character at the end of the CheckSum(10) field.

All messages must begin with the BeginString(8) field and terminate with the CheckSum(10) field.

4.3.6 Components

Application level messages, representing the FIX application layer, can organize a collection of fields into a set commonly referred to as a component or a submessage. These components can contain sub-components.

The FIX tagvalue encoding does not represent component boundaries in the encoding. Any component boundary is lost during encoding. Further, FIX tagvalue encoding does not require the collection of fields to be ordered and does not enforce component boundaries around the fields in the encoding.

4.3.7 Groups and repeating groups

4.3.7.1 General

In ISO/IEC 11404, a FIX repeating group is an array of records. An instance of a repeated record is called a repeating group instance in this document.

It is permissible for fields to be repeated within a repeating group. For example, the following represents a repeating group with two repeating instances delimited by tag 372 (first field in the repeating group):

```
384=2<SOH>372=6<SOH>385=R<SOH>372=7<SOH>385=R<SOH>
```

4.3.7.2 Repeating group name

It is recommended that a repeating group be named XXXGrp, e.g. DividendPeriodGrp.

4.3.7.3 NumInGroup field

In tagvalue encoding, repeating group instances are preceded by a count of the number of instances to follow. The count is serialized as a FIX field with a value of datatype *NumInGroup*, commonly referred to as a NumInGroup field.

It is recommended that NumInGroup fields be named NoXXX, e.g. NoContraBrokers(382).

4.3.7.4 Field sequence within a repeating group

- The NumInGroup field [e.g. NoTradingSessions(386), NoAllocs(78)], which specifies the number of repeating group instances, occurs once for a repeating group and must immediately precede the repeating group instances.
- Fields within repeating groups must be specified in the order that the fields are specified in the message definition.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

4.3.7.5 Field presence within a repeating group

- The NumInGroup field is required and must be larger than zero if the repeating group is required, or if the repeating group is optional and the message contains one or more instances for that repeating group.
- If a repeating group field is specified as required, then it must appear in every instance of that repeating group.
- If a repeating group is used in a message, its first field (after the NumInGroup field) must be populated in each instance of the repeating group. This allows implementations of the protocol to use the first field as the indicator for the start of a new instance within the repeating group.
- The first field listed after the NumInGroup field may be a component or nested repeating group. In this case, the first field is defined as the first field of the component or the NumInGroup field of the nested repeating group. The component or nested repeating group becomes necessary for every instance of the outer repeating group.
- The presence of optional or conditionally required fields may vary across repeating group instances.

4.3.7.6 Nested repeating groups

Repeating groups may be nested within another repeating group. Multiple levels of nesting are allowed. In an encoded message, nested repeating groups are serialized as a depth-first tree traversal. That is, all instances of a nested group of the first top-level group instance are encoded before the second instance of the top-level group, and so forth.

Nesting level	Tag	Field name	Notes
Start Level 1	453	NoPartyIDs	This repeating group is the Parties component in the FIX Standard.
	448	> PartyID	Must always be the first field in the repeating group, and must be provided if NoPartyIDs(453) > 0.
	447	> PartyIDSource	Required if NoPartyIDs(453) > 0.
	452	> PartyRole	Required if NoPartyIDs(453) > 0.
	2376	> PartyRoleQualifier	Optional; not required for each repeating group instance.
Start Level 2	802	**> NoPartySubIDs**	This nested repeating group is the PtysSubGrp component in the FIX Standard.
	523	> > PartySubID	Required if NoPartySubIDs(802) > 0.
	803	> > PartySubIDType	Required if NoPartySubIDs(802) > 0.
End Level 2			
End Level 1			

4.3.7.7 Nested repeating group example

The following is an example of a Parties repeating group with three instances, two of which contain nested PtysSubGrp repeating groups. This example also demonstrates that repeating group instances may be heterogeneous, meaning that the fields present in an instance can vary across instances.

```
NoPartyIDs(453)=3
    PartyID(448)=DEU
    PartyIDSource(447)=B          (Bank Identifier Code (BIC) ISO 9362)
    PartyRole(452)=1              (Executing Firm)
    NoPartySubIDs(802)=1
          PartySubID(523)=A1
          PartySubIDType(803)=10  (Securities account number)
    PartyID(448)=104317
    PartyIDSource(447)=H          (CSD Participant Number)
    PartyRole(452)=83             (Clearing Account)
    PartyID(448)=GSI
```

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

```
PartyIDSource(447)=B          (Bank Identifier Code (BIC) ISO 9362)
PartyRole(452)=4              (Clearing Firm)
PartyRoleQualifier(2376)=23   (Firm or legal entity)
NoPartySubIDs(802)=1
      PartySubID(523)=C3
      PartySubIDType(803)=10  (Securities account number)
```

This example is encoded in FIX tagvalue format as follows:

```
453=3<SOH>448=DEU<SOH>447=B<SOH>452=1<SOH>802=1<SOH>523=A1<SOH>
803=10<SOH>448=104317<SOH>447=H<SOH>452=83<SOH>448=GSI<SOH>447=B<SOH>
452=4<SOH>2376=23<SOH>802=1<SOH>523=C3<SOH>803=10<SOH>
```

4.3.8 Encoded data fields

4.3.8.1 General

Tagvalue encoding provides features for embedding fields in any IANA-registered encoding, possibly using multibyte character sets. Such fields must be preceded by an associated Length field that specifies the number of octets in the encoded data field.

4.3.8.2 MessageEncoding field

MessageEncoding(347) is a field in the StandardHeader component that gives the name of an encoding used in a message.

4.3.8.3 Examples of using encoded data fields for Japanese language support

Example 1 — Specify the English value as Issuer plus Japanese character set as EncodedIssuer(349)

Tag	Field name	Value
...Other standard header fields		
347	MessageEncoding	Shift_JIS
...Other standard header fields		
...Other message body fields		
106	Issuer	HITACHI
348	EncodedIssuerLen	10
349	EncodedIssuer	日立製作所
...Other message body fields		

Example 2 — Specify the English value as Issuer(106) plus Japanese character set as EncodedIssuer(349). Specify the English value as Text(58) plus Japanese character set as EncodedText(357).

Tag	Field name	Value
...Other standard header fields		
347	MessageEncoding	Shift_JIS
...Other standard header fields		
...Other message body fields		
106	Issuer	HITACHI
348	EncodedIssuerLen	10
349	EncodedIssuer	日立製作所
...Other message body fields		
58	Text	This is a test
356	EncodedTextLen	17
357	EncodedText	これはテストです
...Other message body fields		

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

4.3.8.4 Precaution when using multibyte encodings

FIX tagvalue encoding processors must use the Length field associated with the encoded data field when parsing encoded data fields to avoid field truncation and subsequent decoding errors. There is the possibility that one of the octets in a multibyte encoded data field contains the 0x01 value, which can be interpreted by message parsers as the <SOH> field delimiter if the associated Length field is not honoured.

5 Standard header and trailer

5.1 General

Fields specified in the standard header and trailer serve as delimiters of a message. These fields provide features for message integrity, a body length in the header and a checksum in the trailer.

5.2 Standard header

5.2.1 General

Each session or application layer message is preceded by a standard header. The header identifies the message type and length. Higher layers (such as a session layer or an application layer) may extend the definitions of the standard header and trailer by using additional fields, groups or components.

5.2.2 Body length calculation

The message length must be specified in the BodyLength(9) field. The length must be calculated by counting the number of octets in the message following the end-of-field delimiter (<SOH>) of BodyLength(9), up to and including the end-of-field delimiter (<SOH>) of the field immediately preceding the CheckSum(10) field.

5.2.3 Standard header definition

Tag	Field name	Datatype	Required	Comments
8	BeginString	String	Y	FIX.4.2 | FIX.4.4 | FIXT.1.1 BeginString(8) must be the first field in the message.
9	BodyLength	Length	Y	Message length, in octets. BodyLength(9) must be the second field in the message.
35	MsgType	String	Y	Defines message type. MsgType(35) must be the third field in the message.
90	SecureDataLen	Length	N	Length field for SecureData(91). SecureDataLen(90) must be present and unencrypted if SecureData(91) is present in the message.
91	SecureData	Data	N	Encrypted message content. Tag number, separator ("="), and delimiter (<SOH>) must be unencrypted. If present in the message, SecureData(91) must be immediately preceded by SecureDataLen(90).
347	MessageEncoding	String	N	Type of message encoding used in a message's encoded data fields. MessageEncoding(347) must be specified if any fields of datatype *data* are present in the message.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

5.3 Standard trailer

5.3.1 Standard trailer definition

Tag	Field name	Datatype	Required	Comments
93	SignatureLength	Length	N	Length field for Signature(89). SignatureLength(93) must be present and unencrypted if Signature(89) is present in the message. SignatureLength(93) must not be included as part of the encrypted content in SecureData(91).
89	Signature	Data	N	If Signature(89) is present, it must be immediately preceded by SignatureLength(93). Signature(89) must not be included as part of the encrypted content in SecureData(91).
10	CheckSum	String	Y	Three-octet character representation of the modulo 256 checksum. Checksum(10) must be the last field in the message. The end-of-field delimiter (<SOH>) of Checksum(10) serves as the end of message delimiter.

5.3.2 Checksum

The checksum must be calculated by summing the binary value of each octet from the start of the BeginString(8) field up to and including the end-of-field delimiter (<SOH>) of the field immediately preceding the CheckSum(10) field, then transforming this value using a modulo 256.

The calculated modulo 256 checksum must then be encoded as an ISO/IEC 8859-1 three-octet representation of the decimal value. For example, if the result of the modulo 256 of the sum of the value of the fields is 23, the CheckSum(10) field will be encoded as the ISO/IEC 8859-1 string “10=023”. See Annex A for details.

6 FIX tagvalue datatypes

6.1 General

Each field in FIX has a 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.

FIX datatypes are shared by session and application messages in tagvalue encoding.

6.2 Value space

The value space of FIX datatypes is shared among all FIX message encodings. Value space of datatypes is defined using the vocabulary in ISO/IEC 11404.

6.3 Lexical space

6.3.1 General

This specification defines the lexical rules specific to FIX tagvalue encoding. FIX implementations must follow these lexical rules to achieve interoperability.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

6.3.2 Character encoding

With the exception of fields of datatype *data*, tagvalue encoding uses a single-byte character set. By default, the encoding is ISO/IEC 8859-1, Latin alphabet No. 1. Note that the Latin-1 alphabet is an 8-bit code but reserves two ranges for control codes.

By counterparty agreement, a different single-byte character set may be used.

6.3.3 Lexical encoding for FIX datatypes

Table 1 — FIX datatypes tagvalue encoding

Data type	Semantics	Value space (ISO/IEC 11404)	Tagvalue lexical space
int	integer number	integer	Sequence of character digits without commas or decimals and optional sign character (characters "-" and "0" to "9"). The sign character utilizes one octet (i.e. positive int is "99999" while negative int is "-99999"). Note that int values may contain leading zeros (e.g. "00023" = "23").
TagNum	A field's tag number	ordinal	Sequence of character digits without commas or decimals. Value must be positive and may not contain leading zeros.
SeqNum	A message sequence number	ordinal	Sequence of character digits without commas or decimals. Value must be positive.
NumInGroup	The number of entries in a repeating group	size	Sequence of character digits without commas or decimals. Value must be positive. Fields of datatype *NumInGroup* are referred to as NumInGroup fields.
DayOfMonth	Day number within a month (values 1 to 31)	integer range 1..31	Sequence of character digits without commas or decimals (values 1 to 31).
float	All float fields must accommodate up to fifteen significant digits. The number of decimal places used should be a factor of business or market needs and mutual agreement between counterparties.	real	Sequence of character digits with optional decimal point and sign character (characters "-", "0" to "9" and "."); the absence of the decimal point within the string will be interpreted as the float representation of an integer value. Note that float values may contain leading zeros (e.g. "00023.23" = "23.23") and may contain or omit trailing zeros after the decimal point (e.g. "23.0" = "23.0000" = "23" = "23.").
Qty	Either a whole number (no decimal places) of "shares" (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units)	Scaled radix=10	Same as float

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Table 1 *(continued)*

Data type	Semantics	Value space (ISO/IEC 11404)	Tagvalue lexical space
Price	A price. Note the number of decimal places may vary. For certain asset classes prices may be negative values. For example, prices for options strategies can be negative under certain market conditions.	Scaled radix=10	Same as float
PriceOffset	A price offset, which can be mathematically added to a price. Note the number of decimal places may vary and some fields such as LastForwardPoints may be negative.	Scaled radix=10	Same as float
Amt	Typically representing a price times a qty	Scaled radix=10	Same as float
Percentage	A percentage (e.g. 0.05 represents 5% and 0.9525 represents 95.25%). Note the number of decimal places may vary	real	Same as float
char	A single character. All char fields are case sensitive (i.e. m != M)	character repertoire=8859-1 (Latin-1)[a]	Any character except control characters. By default, ISO/IEC 8859-1 (Latin-1). By counterparty agreement, a different character set may be used.
Boolean	Boolean	boolean	'Y' = True/Yes 'N' = False/No
String	Text. All String fields are case sensitive (i.e. "morstatt" != "Morstatt")	characterstring repertoire=8859-1 (Latin-1)[b]	Alphanumeric free-format strings can include any character except control characters.
MultipleCharValue[c]	Set of character codes	set element = character repertoire=8859-1 (Latin-1)[d]	String containing one or more space-delimited single character values, e.g. "2 A F".
MultipleStringValue[e]	Set of string codes	set element = character string repertoire=8859-1 (Latin-1)[f]	String containing one or more space-delimited multiple character values, e.g. "AV AN A".
Country	External code set ISO 3166-1	array element= character index-lowerbound=1 index-upperbound=2	Two-character code
Currency	External code set ISO 4217	array element = character index-lowerbound=1 index-upperbound=3	Three-character code
Exchange	External code set ISO 10383	array element = character index-lowerbound=1 index-upperbound=4	Four-character code

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Table 1 *(continued)*

Data type	Semantics	Value space (ISO/IEC 11404)	Tagvalue lexical space
MonthYear	Month and year of instrument maturity or expiration	characterstring	String representing month of a year. An optional day of the month can be appended or an optional week code. Valid formats: YYYYMM YYYYMMDD YYYYMMWW Valid values: YYYY = 0000-9999; MM = 01-12; DD = 01-31; WW = w1, w2, w3, w4, w5.
UTCTimestamp	UTC date/time	time time-unit = millisecond or up to picosecond by bilateral agreement	String representing time or date combination represented in UTC (Universal Time Coordinated) in either YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss* format, colons, dash and period required. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, MM = 0059, SS = 00-60 (60 only if UTC leap second), sss* fractions of seconds. The fractions of seconds may be empty when no fractions of seconds are conveyed (in such a case the period is not conveyed), it may include three digits to convey milliseconds, six digits to convey microseconds, nine digits to convey nanoseconds, twelve digits to convey picoseconds. Other number of digits may be used with bilateral agreement. 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 31 December or 30 June. The IERS considers 31 March and 30 September 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 https://www.usno.navy.mil/USNO/time/master-clock/leap-seconds). Examples: “20011217-09:30:47.123” milliseconds “20011217-09:30:47.123456” microseconds “20011217-09:30:47.123456789” nanoseconds “20011217-09:30:47.123456789123” picoseconds

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Table 1 *(continued)*

Data type	Semantics	Value space (ISO/IEC 11404)	Tagvalue lexical space
UTCTimeOnly	UTC time of day	time time-unit = millisecond or up to picosecond by bilateral agreement	String representing time-only represented in UTC (Universal Time Coordinated) in either HH:MM:SS (whole seconds) or HH:MM:SS .sss* (milliseconds) format, colons and full stop required. This special-purpose field is paired with UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: HH = 00-23, MM = 00-59, SS = 00-60 (60 only if UTC leap second), sss* fractions of seconds. The fractions of seconds may be empty when no fractions of seconds are conveyed (in such a case the period is not conveyed), it may include three digits to convey milliseconds, six digits to convey microseconds, nine digits to convey nanoseconds, twelve digits to convey picoseconds. Other number of digits may be used with bilateral agreement. Examples: “13:20:00.123” milliseconds “13:20:00.123456” microseconds “13:20:00.123456789” nanoseconds “13:20:00.123456789123” picoseconds
UTCDateOnly	UTC date	time time-unit = day	Date represented in UTC (Universal Time Coordinated) in YYYYMMDD format. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31.
LocalMktDate	Local date	time time-unit = day	Date of local market (as opposed to UTC) in YYYYMMDD format. Valid values: YYYY = 0000-9999, MM = 01-12, DD = 01-31.
TZTimeOnly	Time of day with timezone	time time-unit = millisecond or up to picosecond by bilateral agreement	Time represented based on the ISO 8601 series. This is the time with a UTC offset to allow identification of local time and time zone 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.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-1:2022

© ISO 2022 – All rights reserved

Table 1 *(continued)*

Data type	Semantics	Value space (ISO/IEC 11404)	Tagvalue lexical space
TZTimestamp	Date or time with timezone	time time-unit = millisecond or up to picosecond by bilateral agreement	String representing a time or date combination representing local time with an offset to UTC to allow identification of local time and time zone offset of that time. The representation is based on the ISO 8601 series. 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* fractions of seconds. The fractions of seconds may be empty when no fractions of seconds are conveyed (in such a case the period is not conveyed), it may include three digits to convey milliseconds, six digits to convey microseconds, nine digits to convey nanoseconds, twelve digits to convey picoseconds. Other number of digits may be used with bilateral agreement. Examples: "20060901-07:39Z" is 07:39 UTC on 1st of September 2006 "20060901-02:39-05" is five hours behind UTC, thus Eastern Time on 1st of September 2006 "20060901-15:39+08" is eight hours ahead of UTC, thus Hong Kong/Singapore time on 1st of September 2006 "20060901-13:09+05:30" is 5.5 hours ahead of UTC, thus 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 time zone)
Length[g]	Length of a data field in octets	size	Sequence of character digits without commas or decimals. Value must be positive. Fields of datatype *Length* are referred to as Length fields. The Length field must be associated with a field of datatype data. The Length field must specify the number of octets of the value contained in the associated data field up to but not including the terminating <SOH>.
data[h]	Opaque data or variable-length string	A union of two datatypes: octetstring and characterstring repertoire=(value encoded using the encoding specified in the MessageEncoding(347) field	Raw data with no format or content restrictions, or a character string encoded as specified by MessageEncoding(347). Fields of datatype data must have an associated field of type Length. Fields of datatype data must be immediately preceded by their associated Length field.

STANDARDSISO.COM : Click to view the full PDF of iso 3531-1:2022

© ISO 2022 – All rights reserved