
**Financial services — Financial
information eXchange session layer —
Part 1:
FIX tagvalue encoding**

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO 3531-1:2022](https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/iso-3531-1-2022)

<https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/iso-3531-1-2022>



iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO 3531-1:2022](https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/iso-3531-1-2022)

<https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/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

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

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.

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.

(standards.iteh.ai)

[ISO 3531-1:2022](https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/iso-3531-1-2022)

<https://standards.iteh.ai/catalog/standards/sist/a4c14f8b-f4d6-4e3b-a0ca-6ac067749822/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.

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*;

- 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.

- 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.

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

```