<http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm>

|  |  |
| --- | --- |
| **FIX Sessions** | FIX Antenna .NET Programmer's Guide |

A FIX session is defined as a bi-directional stream of ordered messages between two parties within a continuous sequence number series.

This section will describe the following concepts, related to FIX sessions:

This topic contains the following sections.

* [Introduction and Terminology](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#introduction)
* [Sequence Numbers](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#sequence)
* [Heartbeats](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#heartbeats)
* [Ordered Message Processing](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#ordered)
* [Possible Duplicates](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#dublicates)
* [Possible Resends](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#resends)
* [Data Integrity](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#integrity)
* [Message Acknowledgment](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#acknowledgment)
* [Encryption](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#encryption)
* [See Also](http://corp-web.b2bits.com/fixanet/doc/html/html/60e5d73c-f584-41d1-b6b5-91c233dbd33f.htm#seeAlsoSection)

**Introduction and Terminology**

A single FIX session can exist across multiple sequential (not concurrent) physical connections.

Parties can connect and disconnect multiple times while maintaining a single FIX session. Connecting parties must bi-laterally agree[[1]](#footnote-1) as to when sessions are to be started/stopped based upon individual system and time zone requirements. Resetting the inbound and outbound sequence numbers back to 1[[2]](#footnote-2), for whatever reason, constitutes the beginning of a new FIX session.

It is recommended that a new FIX session be established once within each 24 hour period. It is possible to maintain 24 hour connectivity and establish a new set of sequence numbers by sending a Logon message with the ResetSeqNumFlag set.

The FIX session protocol is based on an optimistic model. Normal delivery of data is assumed (i.e. no communication level acknowledgment of individual messages) with errors in delivery identified by message sequence number gaps. This section provides details on the implementation of the FIX session layer and dealing with message sequence gaps[[3]](#footnote-3).

Valid FIX Message is a message that is properly formed according to this specification and contains a valid body length and checksum field, see [FIX Messages](http://corp-web.b2bits.com/fixanet/doc/html/html/464a7a55-67c0-483d-95e4-18aa083715b1.htm)

*Initiator* establishes the telecommunications link and initiates the session via transmission of the initial Logon message.

*Acceptor* is the receiving party of the FIX session. This party has responsibility to perform first level authentication and formally declare the connection request accepted through transmission of an acknowledgment Logon message.

*Unregistered Acceptor* is the acceptor, which does not exist by the time the new incoming logon message appears.

*FIX Connection* is comprised of three parts: *logon*, *message exchange*, and *logout*.

*FIX Session* is comprised of one or more FIX Connections, meaning that a FIX Session spans multiple logins.

**Sequence Numbers**

All FIX messages are identified by a unique sequence number. Sequence numbers are initialized at the start of each FIX session (see Session Protocol section) starting at 1 (one) and increment throughout the session. Monitoring sequence numbers will enable parties to identify and react to missed messages and to gracefully synchronize applications when reconnecting during a FIX session.

Each session will establish an independent incoming and[[4]](#footnote-4) outgoing sequence series; participants will maintain a sequence series to assign to outgoing messages and a separate series to monitor for sequence gaps on incoming messages.

**Heartbeats**

During periods of message inactivity, FIX applications will generate Heartbeat messages at regular time intervals. The heartbeat monitors the status of the communication link and identifies incoming sequence number gaps. The Heartbeat Interval is declared by the session initiator using the HeartBtInt field in the Logon message. The heartbeat interval timer should be reset after every message is transmitted (not just heartbeats). The HeartBtInt value should be agreed upon by the two firms and specified by the Logon initiator and echoed back by the Logon acceptor.

|  |
| --- |
| Note**Note** |
| Note that the same HeartBtInt value is used by both sides, the Logon "initiator" and Logon "acceptor". |

**Ordered Message Processing**

The FIX protocol assumes complete ordered delivery of messages between parties. Implementers should consider this when designing message gap fill processes. Two options exist for dealing with gaps, either request all messages subsequent to the last message received or ask for the specific message missed while maintaining an ordered list of all newer messages. For example, if the receiver misses the second of five messages, the application could ignore messages 3 through 5 and generate a resend request for messages 2 through 5, or, preferably 2 through 0 (where 0 represents infinity). Another option would involve saving messages 3 through 5 and resending only message 2. In both cases, messages 3 through 5 should not be processed before message 2.

**Possible Duplicates**

When a FIX engine is unsure if a message was successfully received at its intended destination or when responding to a resend request, a possible duplicate message is generated. The message will be a retransmission (with the same sequence number) of the application data in question with the PossDupFlag included and set to "Y" in the header. It is the receiving application's responsibility to handle the message (i.e. treat as a new message or discard as appropriate). All messages created as the result of a resend request will contain the PossDupFlag field set to "Y". Messages lacking the PossDupFlag field or with the PossDupFlag field set to "N" should be treated as original transmissions.

|  |
| --- |
| Note**Note** |
| When retransmitting a message with the PossDupFlag set to Y, it is always necessary to recalculate the CheckSum value. The only fields that can change in a possible duplicate message are the CheckSum, OrigSendingTime, SendingTime, BodyLength and PossDupFlag. Fields related to encryption (SecureDataLen and SecureData) may also require recasting. |

**Possible Resends**

Ambiguous application level messages may be resent with the PossResend flag set. This is useful when an order remains unacknowledged for an inordinate length of time and the end-user suspects it had never been sent. The receiving application must recognize this flag and interrogate internal fields (order number, etc[[5]](#footnote-5).) to determine if this order has been previously received.

|  |
| --- |
| Note**Note** |
| The possible resend message will contain exactly the same body data but will have the PossResend flag and will have a new sequence number. In addition the CheckSum field will require recalculation and fields related to encryption (SecureDataLen and SecureData) may also require recasting. |

**Data Integrity**

The integrity of message data content can be verified in two ways: verification of message length and a simple checksum of characters.

*The message length* is indicated in the BodyLength field and is verified by counting the number of characters in the message following the BodyLength field up to, and including, the delimiter immediately preceding the CheckSum tag ("10=").

*The CheckSum* integrity check is calculated by summing the binary[[6]](#footnote-6) value of each character from the "8" of "8=" up to and including the <SOH> character immediately preceding the CheckSum tag field and comparing the least significant eight bits of the calculated value to the CheckSum value

**Message Acknowledgment**

The FIX session protocol is based on an optimistic model; normal delivery of data is assumed (i.e. no acknowledgment of individual messages) with errors in delivery identified by message sequence number gaps. Each message is identified by a unique sequence number. It is the receiving application's responsibility to monitor incoming sequence numbers to identify message gaps for response with resend request messages.

The FIX protocol does not support individual message acknowledgment. However, a number of application messages require explicit application level acceptance or rejection. Orders, cancel requests, cancel/replace requests, allocations, etc. require specific application level responses, executions can be rejected with the DK message but do not require explicit acceptance.

**Encryption**

The exchange of sensitive data across public carrier networks may make it advisable to employ data encryption techniques to mask the application messages.

The choice of encryption method will be determined by mutual agreement of the two parties involved in the connection.

Any field within a message can be encrypted and included in the SecureData field, however, certain explicitly identified fields must be transmitted unencrypted. The clear (unencrypted) fields can be repeated within the SecureData field to serve as an integrity check of the clear data.

When encryption is employed, it is recommended but not required that all fields within the message body be encrypted. If repeating groups are used within a message and encryption is applied to part of the repeating group, then the entire repeating group must be encrypted.

Embedded in the protocol are fields, which enable the implementation of a public key signature and encryption methodology, straight DES encryption and clear text. T he previously agreed upon encryption methodology is declared in the Logon message. (For more detail on implementation of various encryption techniques see the application notes section on the FIX Web Site.)

**See Also**

#### Other Resources

[Fix Protocol Concepts](http://corp-web.b2bits.com/fixanet/doc/html/html/e0590827-dcd5-4f9c-a6a6-b181068efbe4.htm)

[FixAntenna.NET basic development concepts: Sessions](http://corp-web.b2bits.com/fixanet/doc/html/html/b141ba6a-e0b7-4c0f-b488-a68089aa04e1.htm)

[The FIX Protocol Organization](http://www.fixprotocol.org/)

FIX Antenna .NET | Syntax based on .NET Framework version 4.0

[Copyright B2BITS 2000-2016. All rights reserved.](http://b2bits.com/)

Send comments on this topic to [Feedback e-mail: technical@btobits.com](mailto:technical@btobits.com?Subject=FIX%20Antenna%20.NET%20Programmer's%20Guide:%20FIX%20Sessions&body=Your%20feedback%20is%20used%20to%20improve%20the%20documentation%20and%20the%20product.%20Your%20e-mail%20address%20will%20not%20be%20used%20for%20any%20other%20purpose%20and%20is%20disposed%20of%20after%20the%20issue%20you%20report%20is%20resolved.%20While%20working%20to%20resolve%20the%20issue%20that%20you%20report%2C%20you%20may%20be%20contacted%20via%20e-mail%20to%20get%20further%20details%20or%20clarification%20on%20the%20feedback%20you%20sent.%20After%20the%20issue%20you%20report%20has%20been%20addressed%2C%20you%20may%20receive%20an%20e-mail%20to%20let%20you%20know%20that%20your%20feedback%20has%20been%20addressed.)

1. How? [↑](#footnote-ref-1)
2. How? [↑](#footnote-ref-2)
3. How? – See “Sequence Numbers”. [↑](#footnote-ref-3)
4. Do they match by in-out pairs? [↑](#footnote-ref-4)
5. What else in addition to order number? Perhaps those including the checksum, date/time sent etc… [↑](#footnote-ref-5)
6. Check it exactly. [↑](#footnote-ref-6)