[](http://quickfixn.org/)

[Docs](http://quickfixn.org/tutorial/creating-an-application.html) [Download](http://quickfixn.org/download/) [Get Help](http://quickfixn.org/help/) [About](http://quickfixn.org/about/about-us/)

# Creating An Application

<http://quickfixn.org/tutorial/creating-an-application.html>

Creating a FIX application is easy; simply implement an IApplication:

**public** **class** **MyQuickFixApp** : IApplication

{

**public** **void** **FromApp**(Message msg, SessionID sessionID) { }

**public** **void** **OnCreate**(SessionID sessionID) { }

**public** **void** **OnLogout**(SessionID sessionID) { }

**public** **void** **OnLogon**(SessionID sessionID) { }

**public** **void** **FromAdmin**(Message msg, SessionID sessionID) { }

**public** **void** **ToAdmin**(Message msg, SessionID sessionID) { }

**public** **void** **ToApp**(Message msg, SessionID sessionID) { }

}

These methods will be called on QuickFIX/N events. We’ll explain what each callback does next.

## APPLICATION CALLBACKS

The callbacks in a QuickFIX/N application notify us of events - when a counterparty logs on, when admin messages are sent, and most importantly, when application messages are received.

**FromApp** - every inbound application level message will pass through this method, such as orders, executions, secutiry definitions, and market data.[[1]](#endnote-1)

**FromAdmin** - every inbound admin level message will pass through this method, such as heartbeats, logons, and logouts.

**OnCreate** - this method is called whenever a new session is created.[[2]](#endnote-2)

**OnLogon** - notifies when a successful logon has completed.

**OnLogout** - notifies when a session is offline - either from an exchange of logout messages or network connectivity loss.

**ToApp** - all outbound application level messages pass through this callback before they are sent. If a tag needs to be added to every outgoing message, this is a good place to do that.

**ToAdmin** - all outbound admin level messages pass through this callback.

## BETTER TYPE SAFETY

We **highly recommend** implementing the [Receiving Messages](http://quickfixn.org/tutorial/receiving-messages) tutorial to receive more type safe messages,

## INITIATORS AND ACCEPTORS

QuickFIX/N implements both the initiator and acceptor pattern in FIX.

Initiator is the FIX term for client - we use an Initiator when we are connecting to another party.

Acceptor is the FIX term for server - we use an Acceptor when other parties are connecting to us.[[3]](#endnote-3)

## CREATING OUR APPLICATION

Putting it all together, we implement the Application interface then instantiate an Acceptor:[[4]](#endnote-4)

**using** QuickFix;

**public** **class** **MyQuickFixApp** : IApplication

{

**public** **void** **FromApp**(Message msg, SessionID sessionID) { }

**public** **void** **OnCreate**(SessionID sessionID) { }

**public** **void** **OnLogout**(SessionID sessionID) { }

**public** **void** **OnLogon**(SessionID sessionID) { }

**public** **void** **FromAdmin**(Message msg, SessionID sessionID) { }

**public** **void** **ToAdmin**(Message msg, SessionID sessionID) { }

**public** **void** **ToApp**(Message msg, SessionID sessionID) { }

}

**public** **class** **MyApp**

{

**static** **void** **Main**(**string**[] args)

{

SessionSettings settings = **new** **SessionSettings**(args[0]);

IApplication myApp = **new** **MyQuickFixApp**();

IMessageStoreFactory storeFactory = **new** **FileStoreFactory**(settings);

ILogFactory logFactory = **new** **FileLogFactory**(settings);

ThreadedSocketAcceptor acceptor = **new** **ThreadedSocketAcceptor**(

myApp,

storeFactory,

settings,

logFactory);

acceptor.**Start**();

**while** (**true**)

{

System.Console.**WriteLine**("o hai");

System.Threading.Thread.**Sleep**(1000);

}

acceptor.**Stop**();

}

}

Please view the [Receiving Messages](http://quickfixn.org/tutorial/receiving-messages) tutorial to see how to implement type safe message callbacks. **This is highly recommended.**

Switching this to an Initiator is as simple as swapping out the ThreadedSocketAcceptor class for the SocketInitiator class.[[5]](#endnote-5)

The IMessageStore keeps a record of all outgoing messages for FIX session level messaging.[[6]](#endnote-6) We could implement our own store by implementing the MessageStoreFactory interface.

There are a few options for logging, with file logger probably being the most useful. We could also implement our own logger by implementing the ILogFactory interface.

* [Creating An Application](http://quickfixn.org/tutorial/creating-an-application.html)
* [Sending Messages](http://quickfixn.org/tutorial/sending-messages.html)
* [Receiving Messages](http://quickfixn.org/tutorial/receiving-messages.html)
* [Repeating Groups](http://quickfixn.org/tutorial/repeating-groups.html)
* [Configuration](http://quickfixn.org/tutorial/configuration.html)
* [Example Apps](http://quickfixn.org/tutorial/example-applications.html)
* [Custom Fields, Groups, and Messages](http://quickfixn.org/tutorial/custom-fields-groups-and-messages.html)
* [QuickFIX .NET Wrapper Compatibility](http://quickfixn.org/tutorial/compatibility.html)

# Sending Messages

<http://quickfixn.org/tutorial/sending-messages.html>

Sending FIX messages in QuickFIX/N is simple:

FIX44.NewOrderSingle order = **new** FIX44.**NewOrderSingle**(

**new** **ClOrdID**("1234"),

**new** **Symbol**("AAPL"),

**new** **Side**(Side.BUY),

**new** **TransactTime**(DateTime.Now),

**new** **OrdType**(OrdType.MARKET));

Session.**SendToTarget**(order, sessionID);

First, we need to learn how to direct messages with Sessions.

## QUICKFIX SESSIONS

When sending a message, we must tell QuickFIX which Session to send it to.[[7]](#endnote-7)

All QuickFIX Sessions are identified by fields in the header of a message, usually SenderCompID, TargetCompID, and BeginString. These are specified in the config file.[[8]](#endnote-8)

SenderCompID=CONNAMARA

TargetCompID=CBOE

BeginString=FIX.4.4

There are a few patterns to gather the session. We can grab the SessionID when it is created and cache it:[[9]](#endnote-9)

**private** SessionID MySessionID { **get**; **set**; }

**public** **void** **OnCreate**(SessionID sessionID)

{

MySessionID = sessionID;

}

We can get the SessionID when responding to an incoming message:

**public** **void** **OnMessage**(FIX42.ExecutionReport execution, SessionID sessionID)

{

**ProcessExecution**(execution, sessionID);

}

Or, we can construct a SessionID by matching the values from our config file:

**var** mySessionID = **new** **SessionID**("FIX4.4", "CONNAMARA", "CBOE");

## CREATING AND SENDING A MESSAGE

The preferred constructor to use includes the specific FIX version and message type. We also pass in the required fields:

import QuickFix;

import QuickFix.Fields;

**var** order = **new** QuickFix.FIX44.**NewOrderSingle**(

**new** **ClOrdID**("1234"),

**new** **Symbol**("AAPL"),

**new** **Side**(Side.BUY),

**new** **TransactTime**(DateTime.Now),

**new** **OrdType**(OrdType.LIMIT));

To set fields, use the message’s field properties:

order.Price = **new** **Price**(**new** **decimal**(22.4));

order.Account = **new** **Account**("18861112");

Putting it all together - creating the message, setting its required fields, setting two additional fields, using SessionID from the section above, we send the message on its way:

**var** order = **new** QuickFix.FIX44.**NewOrderSingle**(

**new** **ClOrdID**("1234"),

**new** **Symbol**("AAPL"),

**new** **Side**(Side.BUY),

**new** **TransactTime**(DateTime.Now),

**new** **OrdType**(OrdType.LIMIT));

order.Price = **new** **Price**(**new** **decimal**(22.4));

order.Account = **new** **Account**("18861112");

Session.**SendToTarget**(order, sessionID);

## ALTERNATIVE CONSTRUCTORS AND FIELD SETTERS

The type safe way – the best way – was demonstrated above, but there are a few other ways to create messages and set fields.

Each message type has a default constructor:

**var** order = **new** QuickFix.FIX44.**NewOrderSingle**();

order.ClOrdID = **new** **ClOrdID**("1234");

order.Symbol = **new** **Symbol**("AAPL");

order.Side = **new** **Side**(Side.BUY);

We have the QuickFIX C++ and QuickFIX/J style get/set methods available, which are also type safe:

order.**Set**(**new** **TransactTime**(DateTime.Now));

order.**Set**(**new** **OrdType**(OrdType.LIMIT));

For setting a field that isn’t a property of a message, use setField:

order.**SetField**(**new** **Account**("18861112"));

Here we create base Message class; it has no properties so SetField must be used everywhere. This style is not recommended:

**var** order = **new** QuickFix.**Message**();

order.Header.**SetField**(**new** **MsgType**("D"));

order.**SetField**(**new** **ClOrdID**("1234"));

order.**SetField**(**new** **Symbol**("AAPL"));

order.**SetField**(**new** **Side**(Side.BUY));

order.**SetField**(**new** **TransactTime**(DateTime.Now));

order.**SetField**(**new** **OrdType**(OrdType.LIMIT));

* [Creating An Application](http://quickfixn.org/tutorial/creating-an-application.html)
* [Sending Messages](http://quickfixn.org/tutorial/sending-messages.html)
* [Receiving Messages](http://quickfixn.org/tutorial/receiving-messages.html)
* [Repeating Groups](http://quickfixn.org/tutorial/repeating-groups.html)
* [Configuration](http://quickfixn.org/tutorial/configuration.html)
* [Example Apps](http://quickfixn.org/tutorial/example-applications.html)
* [Custom Fields, Groups, and Messages](http://quickfixn.org/tutorial/custom-fields-groups-and-messages.html)
* [QuickFIX .NET Wrapper Compatibility](http://quickfixn.org/tutorial/compatibility.html)

# Configuring QuickFIX/N

<http://quickfixn.org/tutorial/configuration.html>

An acceptor or initiator can maintain as many FIX sessions as you would like. A FIX session is defined in QuickFix/N as a unique combination of a **BeginString** (the FIX version number), a **SenderCompID** (your ID), and a **TargetCompID** (the ID of your counterparty).[[10]](#endnote-10) A **SessionQualifier** can also be used to disambiguate otherwise identical sessions.

Each session can have several settings associated with them. Some of these settings may not be known at compile time and are therefore passed via a class called SessionSettings. SessionSettings behaves as a data dictionary[[11]](#footnote-2) that can be set and queried.

The SessionSettings class has the ability to parse settings out of any System.IO.TextReader. You can also simply pass it a filename. If you decide to write your own components (eg, storage for a particular database) you may also use this to store settings.

A settings file is set up with two types of heading; [DEFAULT] and [SESSION]. [SESSION] tells QuickFIX/N that a new Session is being defined. [DEFAULT] is a section to define settings which will be associated with sessions that don’t explicitly define them. QuickFIX/N itself will not define default values for all required settings. If you do not provide a setting that QuickFIX/N needs, it will throw a ConfigError telling you what setting is missing or improperly formatted.

Below are the settings that can be associated with a session based on the default functionality provided with QuickFIX/N, followed by an example.

## QUICKFIX SETTINGS

* [Session](http://quickfixn.org/tutorial/configuration.html#session)
* [Validation](http://quickfixn.org/tutorial/configuration.html#validation)
* [Initiator](http://quickfixn.org/tutorial/configuration.html#initiator)
* [Acceptor](http://quickfixn.org/tutorial/configuration.html#acceptor)
* [Storage](http://quickfixn.org/tutorial/configuration.html#storage)
* [File Storage](http://quickfixn.org/tutorial/configuration.html#file)
* [Logging](http://quickfixn.org/tutorial/configuration.html#logging)
* [SSL](http://quickfixn.org/tutorial/configuration.html#ssl)
* [Sample Settings File](http://quickfixn.org/tutorial/configuration.html#sample)

## SESSION

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| BeginString | Version of FIX this session uses | FIXT.1.1  FIX.4.4  FIX.4.3  FIX.4.2  FIX.4.1  FIX.4.0 |  |
| SenderCompID | Your ID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| SenderSubID | (Optional) Your subID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| SenderLocationID | (Optional) Your locationID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| TargetCompID | Counterparty's ID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| TargetSubID | (Optional) Counterparty's subID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| TargetLocationID | (Optional) Counterparty's locationID as associated with this FIX session | case-sensitive alpha-numeric string |  |
| SessionQualifier | Additional qualifier to disambiguate otherwise identical sessions | case-sensitive alpha-numeric string |  |
| DefaultApplVerID | Required only for FIXT 1.1 (and newer). Ignored for earlier transport versions. Specifies the default application version ID for the session. This can be the ApplVerID enum (see the ApplVerID field) or the BeginString for the default version. | FIX.5.0SP2  FIX.5.0SP1  FIX.5.0  FIX.4.4  FIX.4.3  FIX.4.2  FIX.4.1  FIX.4.0 | - |
| ConnectionType | Defines if session will act as an acceptor or initiator | initiator  acceptor | - |
| StartTime | Time of day that this FIX session becomes activated | Time in the format of HH:MM:SS. Uses "TimeZone" if specified, else UTC | - |
| EndTime | Time of day that this FIX session becomes deactivated | Time in the format of HH:MM:SS. Uses "TimeZone" if specified, else UTC | - |
| StartDay | For week long sessions, the starting day of week of the session. Use in combination with StartTime. | Day of week in English using any abbreviation (ie mo, mon, mond, monda, monday are valid) | - |
| EndDay | For week long sessions, the ending day of week for the session. Use in combination with EndTime. | Day of week in English using any abbreviation (ie mo, mon, mond, monda, monday are valid) | - |
| MillisecondsInTimeStamp | Determines if milliseconds should be added to timestamps. Only available for FIX.4.2 and greater. | Y  N | Y |
| ResetOnLogon | Determines if sequence numbers should be reset when receiving a logon request. Acceptors only. | Y  N | N |
| ResetOnLogout | Determines if sequence numbers should be reset to 1 after a normal logout termination. | Y  N | N |
| ResetOnDisconnect | Determines if sequence numbers should be reset to 1 after an abnormal termination | Y  N | N |
| RefreshOnLogon | Determines if session state should be restored from persistence layer when logging on. Useful for creating hot failover sessions. | Y  N | N |
| EnableLastMsgSeqNumProcessed | Add the last message sequence number processed in the header (optional tag 369). | Y  N | N |
| MaxMessagesInResendRequest | Sets the maximum number of messages to retransmit in a single resend request. | Any integer greater than 0 is valid. Use 0 for infinity (default). | 0 |
| SendLogoutBeforeDisconnectFromTimeout | Specifies whether a logout message should be sent before a connection is disconnected due to a timeout. | Y  N | N |
| IgnorePossDupResendRequests | Specifies whether to ignore a resend request when PossDupFlag (tag 43) is set to true | Y  N | N |
| UseLocalTime | Specifies whether to use local machine time for session schedule (StartTime and EndTime). | Y  N | N |
| TimeZone | Specifies time zone ID used for session schedule. Cannot be used with UseLocalTime. Supplied ID will be passed to [TimeZoneInfo.FindSystemTimeZoneById](http://msdn.microsoft.com/en-us/library/system.timezoneinfo.findsystemtimezonebyid.aspx). (See [here](http://stackoverflow.com/questions/7908343/list-of-timezone-ids-for-use-with-findtimezonebyid-in-c) for how to get a list of valid IDs on your system.) | Any time zone ID supported on target system. | - |
| RequiresOrigSendingTime | If N, do not reject SequenceReset/PossDup messages that lack OrigSendingTime | Y  N | Y |

## VALIDATION

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| UseDataDictionary | Tells session whether or not to expect a data dictionary. You should always use a DataDictionary if you are using repeating groups. | Y  N | Y |
| DataDictionary | XML definition file for validating incoming FIX messages. If no DataDictionary is supplied, only basic message validation will be done.  This setting should only be used with FIX transport versions older than FIXT.1.1. See TransportDataDictionary and AppDataDictionary for FIXT.1.1 settings. | Valid XML data dictionary file. QuickFIX/N comes with the following defaults in the spec/fix directory  FIX44.xml  FIX43.xml  FIX42.xml  FIX41.xml  FIX40.xml | - |
| TransportDataDictionary | XML definition file for validating admin (transport) messages. This setting is only valid for FIXT.1.1 (or newer) sessions.  See DataDictionary for older transport versions (FIX.4.0-FIX.4.4) for additional information. | Valid XML data dictionary file. QuickFIX/N comes with the following defaults in the spec/fix directory  FIXT1.1.xml | - |
| AppDataDictionary | XML definition file for validating application messages. This setting is only valid for FIXT.1.1 (or newer) sessions.  See DataDictionary for older transport versions (FIX.4.0-FIX.4.4) for additional information.  This setting supports the possibility of a custom application data dictionary for each session. This setting would only be used with FIXT.1.1 and newer transport protocols. This setting can be used as a prefix to specify multiple application dictionaries for the FIXT transport. For example:  DefaultApplVerID=FIX.4.2 # For default application version ID AppDataDictionary=FIX42.xml # For nondefault application version ID # Use BeginString suffix for app version AppDataDictionary.FIX.4.4=FIX44.xml | Valid XML data dictionary file, QuickFIX/N comes with the following defaults in the spec/fix directory  FIX50SP2.xml  FIX50SP1.xml  FIX50.xml  FIX44.xml  FIX43.xml  FIX42.xml  FIX41.xml  FIX40.xml | - |
| ValidateFieldsOutOfOrder | If set to N, fields that are out of order (ie, body fields in the header, or header fields in the body) will not be rejected. Useful for connecting to systems which do not properly order fields. | Y  N | Y |
| ValidateFieldsHaveValues | If set to N, fields without values (empty) will not be rejected. Useful for connecting to systems that improperly send empty tags. | Y  N | Y |
| ValidateUserDefinedFields | If set to N, user defined fields will not be rejected if they are not defined in the data dictionary, or are not present in messages they do not belong to. | Y  N | Y |
| CheckLatency | If Y, messages must be received from the counterparty within a defined number of seconds (see MaxLatency). It is useful to turn this off if a system uses localtime for its timestamps instead of GMT. | Y  N | Y |
| MaxLatency | If CheckLatency=Y, this defines the number of seconds latency allowed for a message to be processed. | positive integer | 120 |

## INITIATOR

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| ReconnectInterval | Time between reconnection attempts in seconds. Only used for initiators. **NOTE: must be defined in DEFAULT section** | Positive integer | 30 |
| HeartBtInt | Heartbeat interval in seconds. Only used for initiators. | Positive integer | - |
| LogonTimeout | Number of seconds to wait for a logon response before disconnecting. | Positive integer | 10 |
| LogoutTimeout | Number of seconds to wait for a logout response before disconnecting. | Positive integer | 2 |
| SocketConnectPort | Socket port for connecting to a session. Only used for initiators. | Positive integer | - |
| SocketConnectHost | Host to connect to. Only used for initiators. | Valid IP address in the format of x.x.x.x or a domain name | - |
| SocketConnectPort<n> | Alternate socket ports for connecting to a session for failover, where **n** is a positive integer. ie, SocketConnectPort1, SocketConnectPort2... must be consecutive and have a matching SocketConnectHost<n> | Positive integer | - |
| SocketConnectHost<n> | Alternate socket hosts for connecting to a session for failover, where **n** is a positive integer. ie, SocketConnectHost1, SocketConnectHost2... must be consecutive and have a matching SocketConnectPort<n> | Valid IP address in the format of x.x.x.x or a domain name | - |
| SocketNodelay | Disable Nagle's algorithm for this connection. Written data to the network is not buffered pending acknowledgement of previously written data. Currently, this must be defined in the [DEFAULT] section. | Y  N | Y |

## ACCEPTOR

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| SocketAcceptPort | Socket port for listening to incoming connections. Only used with acceptors. | Positive integer, valid open socket port | - |
| SocketAcceptHost | Socket host for listening to incoming connections. If not provided, the acceptor will listen on all network interfaces (0.0.0.0) | Valid IP address in the format of x.x.x.x | 0.0.0.0 |
| SocketNodelay | Disable Nagle's algorithm for this connection. Written data to the network is not buffered pending acknowledgement of previously written data. Currently, this must be defined in the [DEFAULT] section. | Y  N | Y |

## STORAGE

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| PersistMessages | If set to N, no messages will be persisted. This will force QuickFIX to always send GapFills instead of resending messages. Use this if you know you never want to resend a message. Useful for market data streams. | Y  N | Y |

## FILE STORAGE

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| FileStorePath | Directory to store sequence number and message files. | Valid directory for storing files, must have write access. | - |

## LOGGING

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| FileLogPath | Directory to store logs. | Valid directory for storing files, must have write access | - |
| DebugFileLogPath | Directory to store ThreadedClientAcceptor thread logs. | Valid directory for storing files, must have write access | Value of FileLogPath if present, else "log". |

## SSL

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Description** | **Valid Values** | **Default** |
| SSLEnable | Determine if SSL should be used. | Y  N | Y if SSLCertificatePath is specified, otherwise N |
| SSLServerName | The expected certificate name of the server (usually same as DNS name). Only used for initiators. | string | defaults to value of SocketConnectHost |
| SSLProtocols | Specifies the SSLProtocol to use (according to the C# SslProtocols enum). If .NET 4.5 is used there are more valid values. Refer to [this MSDN page](http://msdn.microsoft.com/en-us/library/system.security.authentication.sslprotocols.aspx) for a complete list of valid values. | Ssl2  Ssl3  Tls  Default  Any other valid SslProtocols enum value | Default |
| SSLValidateCertificates | If Y, the remote party's certificate will be verified against the certificate specified by the SSLCACertificate setting (or the operating system's list of trusted CAs if that setting is not specified). **NOTE:** setting the value to N is a security risk. **Setting the value to N will also set SSLCheckCertificateRevocation to N.** | Y  N | Y |
| SSLCheckCertificateRevocation | Determine if certificate revocation list (CRL) should be used to check if SSL certificates have been revoked. Will be overridden to N if SSLValidateCertificates=N. | Y  N | Y |
| SSLCertificate | Specifies which SSL certificate (containing a private key) to use. Required for acceptor, but not for initiator unless client certificates are used. The certificate can be loaded either from a pfx file or from the current user's personal certificate store. | * Path to a .pfx-file to which read access is granted * Distinguished name (ex: CN=CertName) of a certificate in the current user's personal certificate store * Subject name (ex: CertName) of a certificate in the current user's personal certificate store | - |
| SSLCertificatePassword | Password for the SSL certificate | string | - |
| SSLRequireClientCertificate | Determine if acceptor should require client certificates from all acceptors | Y  N | Y |
| SSLCACertificate | Specifies the CA certificate used by acceptor to validate client certificates. If not specified, then all certificates installed on the computer's certificate store (under "Trusted Root Certificates") are used. The certificate can be loaded either from a file or from the current user's personal certificate store. | * Path to a .cer- or .pfx-file to which read access is granted * Distinguished name (ex: CN=CertName) of a certificate in the current user's personal certificate store * Subject name (ex: CertName) of a certificate in the current user's personal certificate store | - |

## SAMPLE INITIATOR SETTINGS FILE

Here is a typical initiator settings file you might find in a firm that wants to connect to several ECNs.

# default settings for sessions

[DEFAULT]

FileStorePath=store

FileLogPath=log

ConnectionType=initiator

ReconnectInterval=60

SenderCompID=TW

# session definition

[SESSION]

# inherit FileStorePath, FileLogPath, ConnectionType,

# ReconnectInterval and SenderCompID from default

BeginString=FIX.4.1

TargetCompID=ARCA

StartTime=12:30:00

EndTime=23:30:00

HeartBtInt=20

SocketConnectPort=9823

SocketConnectHost=123.123.123.123

DataDictionary=somewhere/FIX41.xml

[SESSION]

BeginString=FIX.4.0

TargetCompID=ISLD

StartTime=12:00:00

EndTime=23:00:00

HeartBtInt=30

SocketConnectPort=8323

SocketConnectHost=23.23.23.23

DataDictionary=somewhere/FIX40.xml

[SESSION]

BeginString=FIX.4.2

TargetCompID=INCA

StartTime=12:30:00

EndTime=21:30:00

# overide default setting for RecconnectInterval

ReconnectInterval=30

HeartBtInt=30

SocketConnectPort=6523

SocketConnectHost=3.3.3.3

# (optional) alternate connection ports

# and hosts to cycle through on failover

SocketConnectPort1=8392

SocketConnectHost1=8.8.8.8

SocketConnectPort2=2932

SocketConnectHost2=12.12.12.12

DataDictionary=somewhere/FIX42.xml

## SAMPLE ACCEPTOR SETTINGS FILE

Here is a typical acceptor settings file.

# default settings for sessions

[DEFAULT]

FileStorePath=store

FileLogPath=log

ConnectionType=acceptor

ReconnectInterval=60

SenderCompID=ARCA

# session definition

[SESSION]

# inherit FileStorePath, FileLogPath, ConnectionType,

# ReconnectInterval and SenderCompID from default

BeginString=FIX.4.1

TargetCompID=TW

StartTime=12:30:00

EndTime=23:30:00

HeartBtInt=20

SocketAcceptPort=9823

DataDictionary=somewhere/FIX41.xml

[SESSION]

BeginString=FIX.4.0

TargetCompID=TW

StartTime=12:00:00

EndTime=23:00:00

HeartBtInt=30

SocketAcceptPort=8323

DataDictionary=somewhere/FIX40.xml

[SESSION]

BeginString=FIX.4.2

TargetCompID=TW

StartTime=12:30:00

EndTime=21:30:00

# overide default setting for RecconnectInterval

ReconnectInterval=30

HeartBtInt=30

SocketAcceptPort=6523

# (optional) only listen for incoming connections on a specific host

SocketAcceptHost=127.0.0.1

DataDictionary=somewhere/FIX42.xml

* [Creating An Application](http://quickfixn.org/tutorial/creating-an-application.html)
* [Sending Messages](http://quickfixn.org/tutorial/sending-messages.html)
* [Receiving Messages](http://quickfixn.org/tutorial/receiving-messages.html)
* [Repeating Groups](http://quickfixn.org/tutorial/repeating-groups.html)
* [Configuration](http://quickfixn.org/tutorial/configuration.html)
* [Example Apps](http://quickfixn.org/tutorial/example-applications.html)
* [Custom Fields, Groups, and Messages](http://quickfixn.org/tutorial/custom-fields-groups-and-messages.html)
* [QuickFIX .NET Wrapper Compatibility](http://quickfixn.org/tutorial/compatibility.html)

1. Bookmark [↑](#endnote-ref-1)
2. Bookmark [↑](#endnote-ref-2)
3. Bookmark [↑](#endnote-ref-3)
4. Bookmark [↑](#endnote-ref-4)
5. Bookmark [↑](#endnote-ref-5)
6. Bookmark [↑](#endnote-ref-6)
7. Bookmark [↑](#endnote-ref-7)
8. Bookmark [↑](#endnote-ref-8)
9. Bookmark [↑](#endnote-ref-9)
10. Bookmark [↑](#endnote-ref-10)
11. Where is it? [↑](#footnote-ref-2)