# TCP communication with FIX initiator

## Introduction

* Software for sending FIX messages (“FIX initiator” in the sequel), is decoupled from the end user. It is accessible via a TCP server that acts as a wrapper around the FIX initiator.
* Server communicates with a Client through an exchange of binary data packets. The structure of each particular packet for various scenarios will be explained in the paragraph “Message format” at the end of this text. The packets are built as groups of ascii-based data, delimited by a zero byte (one byte with the value of 0). In all the message format descriptions, the zero byte is represented by the **0** character.
* Server exposes 2 kinds of services:

1. Remote method invocation (RMI)
2. Event emitting

* To consume any kind of service, a server client needs to:

1. Send correct credentials (username and password)
2. Send a correct **Secret key**. (Client receives the Secret key on success of step 1)

* Once the above 2 preconditions are met, the Server creates a session that uniquely determines the connection between the client and the FIX initiator. Session is represented by the **Secret key**. The Server will disallow multiple connections to a single session.

## Communication with TCP server

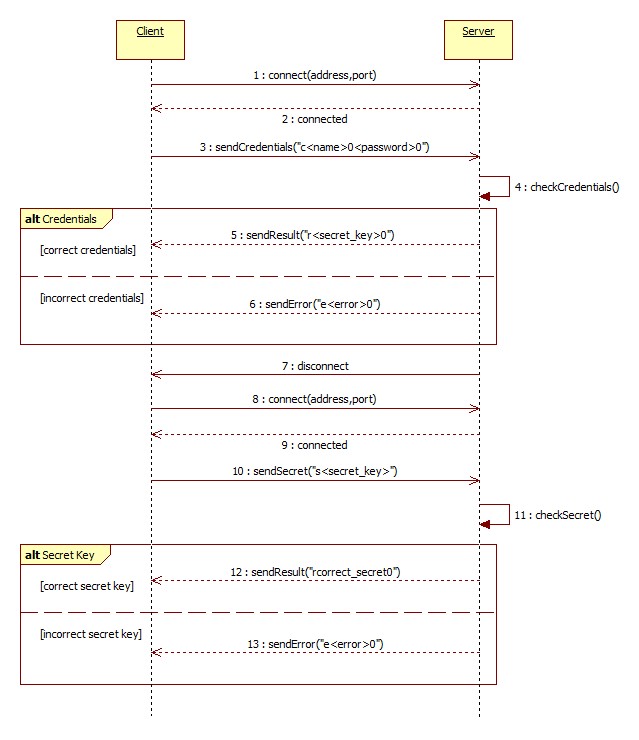
### Successful communication passes 2 stages:

1. Establishing a connection
2. Method invocation / Receiving data (event listening)

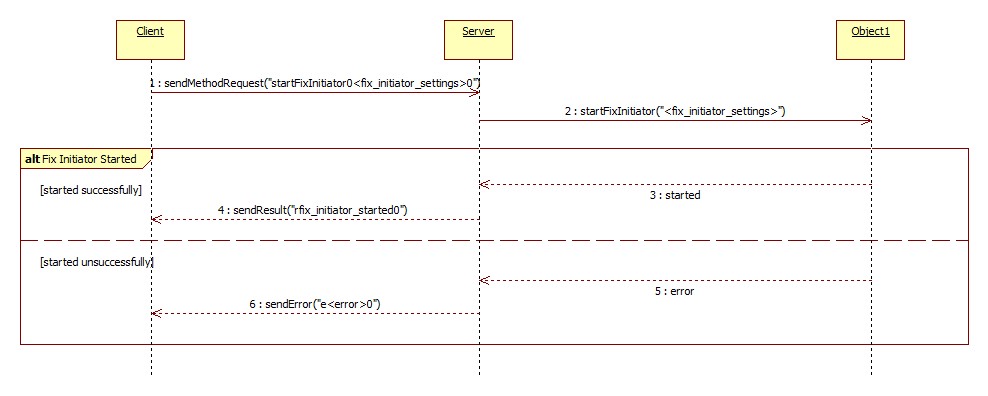
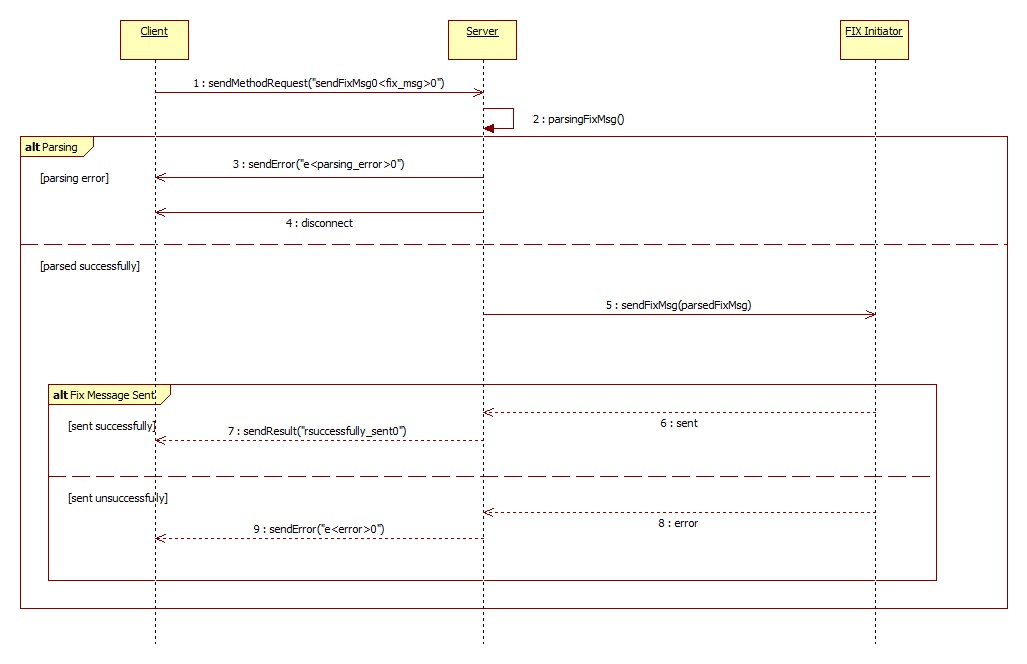
## 1. Successful scenario for establishing a connection

1. Client establishes a *plain* connection (connect a socket) with TCP server
2. Client sends the [credentials](#credentials).
3. Server checks the credentials - with success.
4. On successful check, the Server replies with a randomly generated 16-byte packet. This packet is the **Secret key**. Reply format is: **r<secret\_key>0**
5. Server closes the connection.
6. Client establishes a *plain* connection (connect a socket) with TCP server
7. Client sends the [secret key](#secret).
8. Server checks the secret key - with success.
9. On successful check the Server replies with **rcorrect\_secret0**. After this, the connection is established.
10. Client recieves **rcorrect\_secret0**. After this, client is free to invoke methods and receive data.

## Erroneous scenarios in establishing the connection

* 3a.1. Server checks credentials - without success.
* 3a.2. Server replies with an [error message](#error).
* 3a.3. Server closes the connection.
* 8a.1. Server checks secret key - without success.
* 8a.2. Server replies with an [error message](#error).
* 8a.3. Server closes the connection.
* 

## [Method invocation](#method) and [recieving data](#event)

* Server implement these methods:
* ***startFixInitiator*** – accepts 1 parameter ([FIX initiator settings](#settings)) which represents settings for starting FIX initiator. If the FIX initiator is successfully started, server replies with **rfix\_initiator\_started0**, or else server replies with [error message](#error).   
  
* ***sendFixMsg*** – accepts 1 parameter ([FIX message](#fixmsg)) which represents the FIX message to be sent to the FIX acceptor. Message is being parsed. If the parsing fails, the Server replies with an [invalid structure message error](#error). Upon successful parsing, the message is sent over to the FIX acceptor. If the message is successfully sent, the Server replies with **rsuccessfully\_sent0;** otherwise the Server replies with an [error message](#error).   
  
* ***getStatistics*** – no parameters. If the method is successfully executed, server replies with a [Statistics message](#statistics); otherwise, the Server replies with an [error message](#error).
* ***Events emitted*** by the server:
* ***connectionEstablished –*** sends 1 argument ([Session message](#session)) which represents session between FIX initiator and FIX acceptor
* ***acceptFixMsg –*** sends 1 argument ([FIX message](#fixmsg)) which represents FIX message that is received from the FIX acceptor
* ***connectionClosed –*** sends 1 argument ([Session message](#session)) which represents session that was closed between FIX initiator and FIX acceptor

### Message format

* Credentials  
  **c<username>0<password>0**
* Secret key  
  **s<16 random bytes>**
* Method invocation  
  **<method\_name>0<parameter\_0>0<parameter\_1>0...<parameter\_n>0**
* Server result  
  **r<result>0**
* Server error  
  **e<error>0**
* Server notification  
  **n<notification>0**
* Server event  
  **o<event\_name>0<argument\_1>0<argument\_2>...<argument\_n>0**
* Tag-value sequence  
  **<tag0>0<value0>0...<tag*n*>0<value>0**
* FIX message  
  **<header>0<tags>0**where **header** is a[tag-value sequence](#tagvaluesequence)and **tags** is a[tag-value sequence](#tagvaluesequence)
* Session message  
  **<sessionID tag-value sequence>0**
* Statistics message  
  TODO
* FIX intiator settings  
  **[DEFAULT]\n ReconnectInterval=60\n RefreshOnLogon=Y\n SendRedundantResendRequests=Y\n PersistMessages=Y\n FileStorePath=./data/1\n FileLogPath=./log/1\n HttpAcceptPort=9011\n\n   
  [SESSION]\n ConnectionType=acceptor\n SenderCompID=ELECTRONIFIE\n TargetCompID=NODEQUICKFIX\n BeginString=FIX.4.4\n StartTime=00:00:00\n EndTime=23:59:59\n HeartBtInt=30\n SocketAcceptPort=3223\n UseDataDictionary=Y\n DataDictionary=./fix\_spec/FIX44.xml**