

Generic file formats:

- **ASCII CSV files – bar by bar** (<contract>.csv)

```

2011.12.07 11:50:00 1262.5 1262.5 1262.25 1262.5 360
2011.12.07 11:51:00 1262.5 1262.5 1261.75 1262.25 673
2011.12.07 11:52:00 1262.0 1262.5 1261.75 1262.5 513
  
```

Record delimiter: new line

Field delimiter: space

Fields: <date> <time> <open> <high> <low> <close> <volume>

date is given in YYYY.MM.DD

time is given in HH:MM:SS

- **ASCII CSV files – tick by tick** (<contract>.ticks)

```

2011.09.16 13:03:14 1201.25 0
2011.09.16 13:03:15 1201.0 35
  
```

Record delimiter: new line

Field delimiter: space

Fields: <date> <time> <price> <volume>

date is given in YYYY.MM.DD

time is given in HH:MM:SS

TCP stream formats:

- **Port 9595**

```
*!EUR_CASH_IDEALPRO_USD:pv:20111208:722:1.34075|0!*!COIL_FUT_20120127_IPE_USD:pv:20111208:722:110.13|1!*!EUR_CASH_IDEALPRO_USD:pv:20111208:722:1.34075|0!EUR_CASH_IDEALPRO_USD:pv:20111208:722:1.3407|0!EUR_CASH_IDEALPRO_USD:pv:20111208:722:1.3407|0!
```

Record delimiter: '!'

Heartbeat every 5 seconds: '*'

Fields: <contract>:pv:<date>:<time>:<price>|<volume>

date format is YYYYMMDD, time format is HM without preceding zeroes

- **Port 9191**

```
*
YI_FUT_20111223_NYSELIFFE_USD;35;bid;32.494;1323325719238
YI_FUT_20111223_NYSELIFFE_USD;35;ask;32.525;1323325719238
ES_FUT_20111223_GLOBEX_USD;38;last;1262.5;1323325720518
```

Record delimiter: new line

Field delimiter: ';'

Heartbeat every 5 seconds: '*'

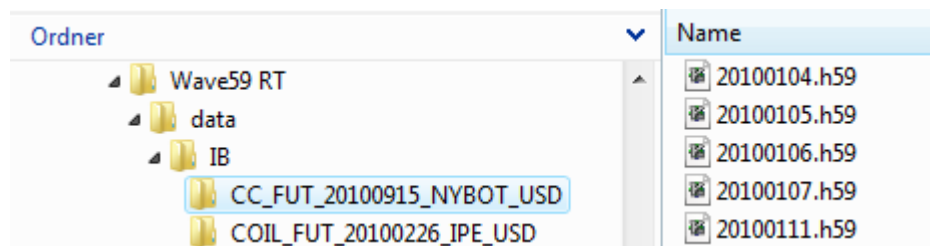
Fields: <contract>;<id>;<type>;<price>;<timestamp>

id is request id internally used by TWS

type is 'bid', 'ask', 'last', 'high', 'low', 'close', 'open', 'low13weeks', 'high13weeks', 'low26weeks', 'high26weeks', 'low52weeks', 'high52weeks', 'MarkPrice'

timestamp is given in milliseconds counted from 1. Jan. 1970 00:00:00.000

Wave59 directory and fileformat:



There's one dedicated directory for each single contract, named by its contract name.

Files:

- <date>.h59

collected bar data for that day, minute by minute

date is given in YYYYMMDD

```
516 1.34165 1.34165 1.34165 1.34165 0
```

```
517 1.34165 1.34165 1.3416 1.3416 7
```

fields: <time> <open> <high> <low> <close> <volume>

- <date>.txt

collected ticks for the current day

date is given in YYYYMMDD

```
516 1.34165
```

```
516 1.34165
```

fields: <time> <price>

- <date>.v.txt

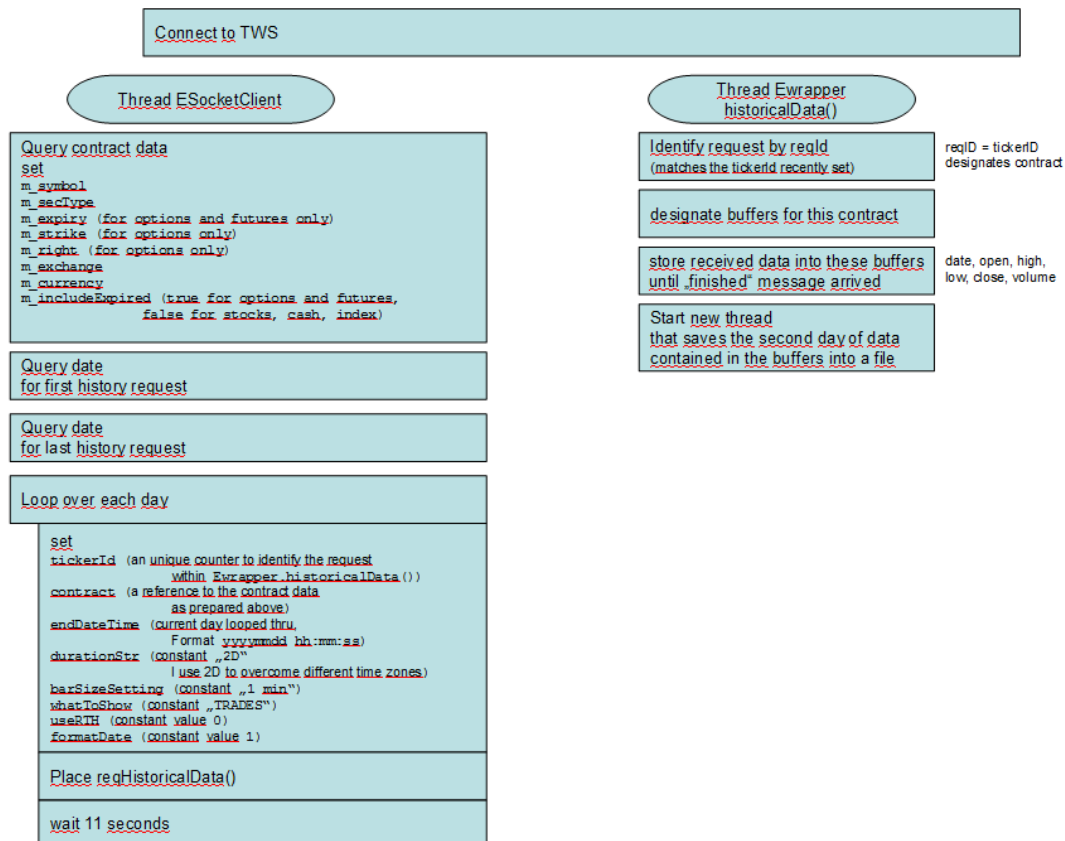
collected volume ticks for the current day

date is given in YYYYMMDD

```
516 456
```

fields: <time> <volume>

reqHistoricalData() is implemented this way:



Linking to TWS using the TWS Java API

Recommended reading:

<http://www.interactivebrokers.com/download/JavaAPIGettingStarted.pdf>

The most important things to remember here are that the `SpreadIBFrame.java` class implements the **EWrapper** interface, which is the part of IB's TWS Java API that defines the methods that receive messages from TWS, and calls the methods in **EClientSocket**, which are used to send messages to TWS.

If you have the skill and confidence to handle Java on your own, you can build your own Java application to link to TWS, using the following steps as a guide.

- 1 Import **com.ib.client.*** into your source code file. This is the package that contains the TWS Java API classes and methods.
- 2 Implement the **EWrapper** interface. This class will receive messages from the socket.
- 3 Override the following methods:

EWrapper Method	Description
tickPrice()	Handles market data.
tickSize()	
tickOptionComputation()	
tickGeneric()	
tickString()	
tickEFP()	
historicalData()	Receives historical data results.

4 Instantiate the **EClientSocket** class. This object will be used to send messages to TWS.

5 Call the following EClientSocket methods:

EClientSocket Method	Description
eConnect()	Connects to TWS.
eDisconnect()	Disconnects from TWS.
reqMktData()	Requests market data.
cancelMktData()	Cancels market data.
reqHistoricalData()	Requests historical data.
cancelHistoricalData()	Cancels historical data.