MT4 Terminal Java API Functional Description

Table of Contents

MT4 Terminal Java API Functional Description....1
Terminology....1
Overview....2
Sequence Diagrams....3
Terminal Server....6

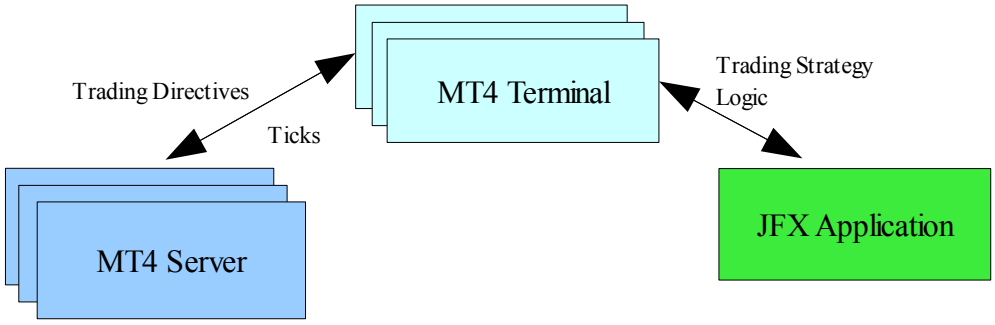
Terminology

Term	Description
Expert Advisor (EA)	Expert Advisors render the trade process management automatic and are perfectly suitable for implementing your own trade strategies. EA life-circle is managed by MT4 Terminal.
JFX	MT4 Terminal Java API
JFX Server	Socket server, part of JFX API, which accepts MT4 Terminal connections and instantiates JFX Strategies
JFX Strategy	Custom trading strategy implemented with use of JFX API. Normally it is represented by java class, which extends *com.jfx.strategy.Strategy* API class.
MT4 Server	Dealing center software from MetaQuotes Software Corp. The server gets all traders' queries for quotations and news, fulfillment of trading operations, placing orders and executing them.
MT4 Terminal	MetaTrader 4 client terminal from MetaQuotes Software Corp., it has been created to provide trade operations and technical analysis in real time mode, when working at Forex, CFD, Futures financial markets. A wide range of orders allows flexible management of trading activities.

Overview

MT4 Terminal Java API (JFX) is intended to provide Java interface to MetaQuotes trading servers through the standard MetaTrader 4 client terminal (MT4 Terminal). It is also intended to fully eliminate the need of MQL4 language in developing of custom trading strategies (expert advisors).

The role of MT4 Terminal is to be an agent between MT4 Server and JFX Application:



Single JFX Application can serve multiple MT4 Terminal's sessions simultaneously.

JFX API consists of three parts:

- JFX MQL4 Expert Advisor (MT4 Terminal's part)
- JFX Communication DLL (MT4 Terminal's part)
- JFX Java API itself (Java Application's part)

To make use of JFX API, one must create its own strategy java class, extending *com.jfx.strategy.Strategy* and overriding *coordinate()* method:

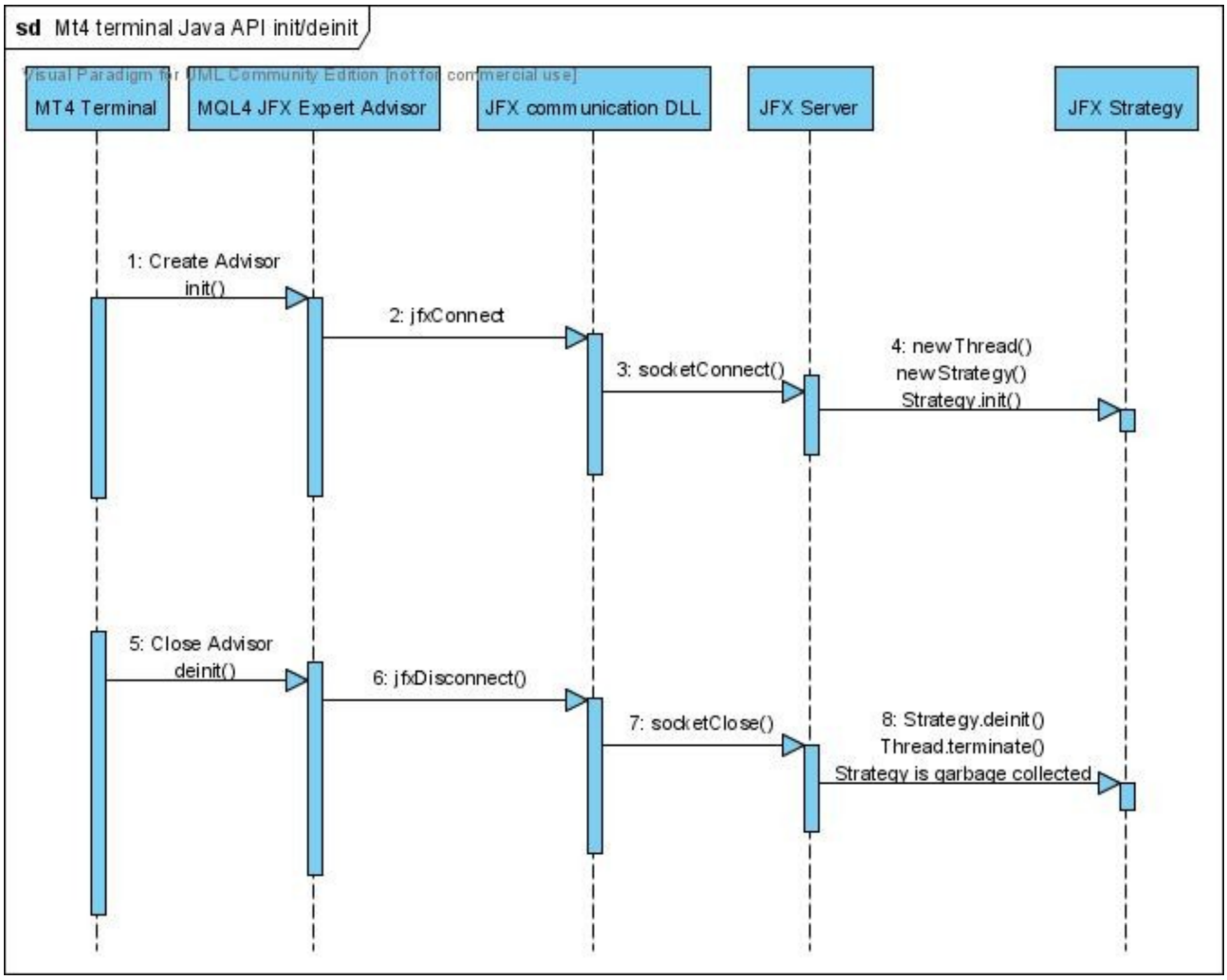
```
public class MyStrategy extends com.jfx.strategy.Strategy {
   public void init(String symbol, int period, StrategyRunner strategyRunner) {
      super.init(symbol, period, strategyRunner);
      //
      // load existing orders, recover itself from the previous shutdown
      //
   }
   public void deinit() {
      // release resources on EA exit
   }
   public void coordinate() {
      // trading logic goes here
      /* make use of all API methods: accountBalance, accountCompany, accountCredit, accountCurrency, accountEquity,
accountFreeMargin, accountMargin, accountName, accountNumber, accountProfit, comment, day, dayOfWeek, dayOfYear,
getLastError, getTickCount, hour, iAC, iAD, iADX, iAlligator, iAO, iATR, iBands, iBars, iBarShift, iBearsPower, iBullsPower,
iBWMFI, iCCI, iClose, iCustom, iDeMarker, iEnvelopes, iForce, iFractals, iGator, iHigh, iHighest, iLow, iLowest, iMA, iMACD,
iMFI, iMomentum, iOBV, iOpen, iOsMA, iRSI, iRVI, iSAR, isConnected, isDemo, iStdDev, isTesting, iStochastic,
isTradeContextBusy, isVisualMode, iTime, iVolume, iWPR, marketInfo, minute, month, objectCreate, objectCreate, objectCreate,
objectDelete, objectGet, objectGetFiboDescription, objectSet, objectSetFiboDescription, objectSetText, objectsTotal, objectType,
orderClose, orderCloseBy, orderClosePrice, orderCloseTime, orderComment, orderCommission, orderDelete, orderExpiration,
orderLots, orderMagicNumber, orderModify, orderOpenPrice, orderOpenTime, orderPrint, orderProfit, orderSelect, orderSend,
```

```
ordersHistoryTotal, orderStopLoss, ordersTotal, orderSwap, orderSymbol, orderTakeProfit, orderTicket, orderType, print,
refreshRates, seconds, timeCurrent, year
        */
    }
}
```

Sequence Diagrams

Name of custom strategy java class is passed to JFX Expert Advisor by MT4 Terminal in order to instantiate it in JFX Application.

Here is an example of initialization/de-initialization process of JFX Application:



JFX Application initialization

1: Create Advisor – when JFX MQL4 Expert Advisor is created, MT4 Terminal calls its init() method

2: jfxConnect – EA init() method connects to Java JFX Application using MT4IF.DLL

3: socketConnect – MT4IF.DLL establishes socket connection to JFX Application and returns session id to EA

4: new Thread/Strategy – at connection time, JFX Server creates new instance of the specified Strategy java class and dedicated separate Thread for its execution.

JFX Application de-initialization

5: Close Advisor – when MT4 terminal removes JFX EA from instrument graph, special deinit() EA's method is invoked

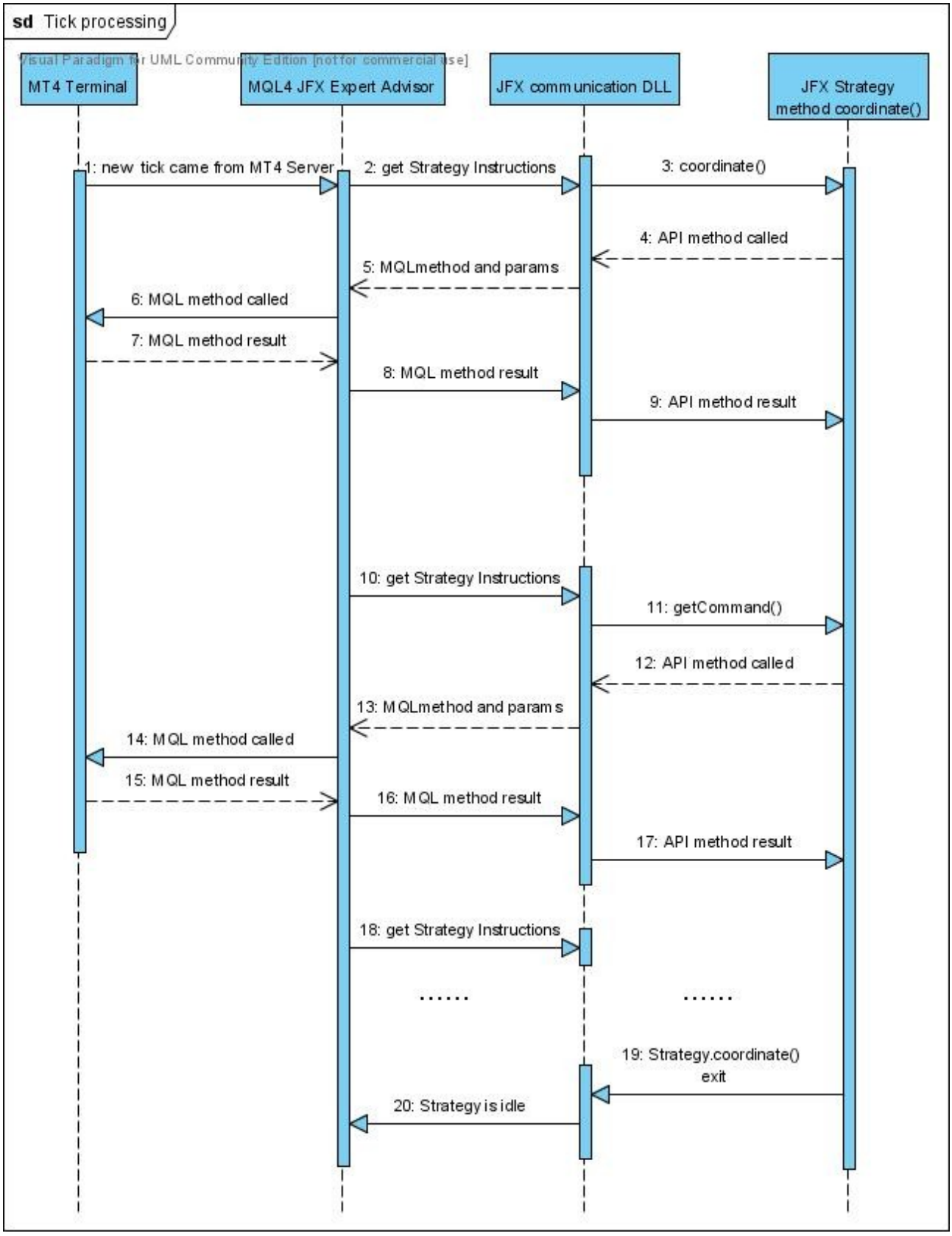
6: jfxDisconnect() - EA's deinit() method drops JFX Application connection using MT4IF.DLL call

7: socketClose() - MT4IF.DLL closes socket connection to JFX Application and releases dedicated session id.

8: Strategy.deinit() - instance of strategy java class is released, Thread is terminated.

JFX Application de-initialization can also occur in case of network connection between MT4 Terminal and JFX Application is broken. In this case MT4IF.DLL tries to re-establish socket connection automatically and for JFX Application it will look like standard initialization process when connection is alive again.

When JFX EA is initialized, most of work is performed at MT4 Server ticks processing:



Terminal Server

Terminal Server is Windows application, which facilitates maintenance of multiple MT4 Terminals, i. e. makes it possible for JFX API client to create simultaneous connections to different brokers with different credentials.