Developer's Guide

Table of Contents

Table of Contents

Introduction and Getting Started 1

1.1. What is NDAXCore™?

NDAXCore™ by Trimtab is new high performance exchange engine, which is programmatically customizable, allowing developers to create their own custom stock, forex, futures, cryptocurrency (Bitcoin or Litecoin), or other Digital Assets

Hosting your own exchange allows you to:

- list securities;
- set your own rules and regulations;
- set your own pricing;
- create and administer your own trading members;
- manage your own record keeping and reporting;
- perform your own market-making;
- sell your own real-time and historic data;
- partner with other exchanges and perform any function that is legally permitted (please note that certain real exchanges may be regulated and may require registration).

You can also program your own logic to prevent flash crashes, institute your own trading curbs, or prevent trading behavior that goes against your exchange's rules.

1.2. About this Guide

This Guide helps to understand the structure of the NDAXCore™ source code. It contains the information about the classes and class relations, database structure, and the example of client application.

Developers' Guide for NDAXCore consists of the following sections:

- **Chapter 1 introduces exchange and this Guide.**
- **Chapter 2** describes the client implementation, structure of the project and its groups.

- **Chapter 3** describes administrator and user applications.
- In **Subject Index** you can find the alphabetically sorted list of the features and elements of NDAXCore and quickly navigate to the pages with information about these elements.

1.3. Tips for Using this Guide

There are two ways to guickly navigate within this Developer's Guide:

- **Table of Contents** (page 3).
- **Subject Index** (page 32), if you want to find information about specific element by its name.

This Guide uses the following notes to emphasize important information about NDAXCore:

Hint — Provides information that may help you to use the information from the Guide effectively

Alert — Alerts you of important information that you need to know when you perform a particular procedure

The names of the project groups, classes and components are marked with bold (e.g., **CachedSymbol**).

1.4. Frequently asked questions

Q. What programming languages are used?

A. C# to customize, compile, and install

Q. Which third party libraries are used in NDAXCore?

A. NLog, log4net, Newtonsoft JSON, and SuperWebSocket.

Q. How it Trader work on NDAXCore?

A. Yes, the example allows a trader to login with a username and password, allows the account info to be updated for testing purposes (buying power, cash, etc.), and allows the trader to place trades with fields such as order type (OCO, LMT, GTC, etc.), quantity, side (buy/sell), type (equity, forex, etc.), position effect, symbol, price, stop price and other fields. The examples also contain grid controls which update in real time.

Q. How to run an administrator program?

A. WinForm-based admin panel, which allows you to create users, manage and delete accounts, view active users, view a log and more.

Q. How is Data stored?

A. Data is saved and loaded to and from an encrypted database. MS SQL is supported by default but can be changed to MySQL or Oracle. Upon startup, Exchange loads all execution reports into memory to initialize the exchange. Exchange updates tables in the database for every transaction, so the database always contains actual live data. Chapter 2.4 contains detailed information about the database tables

1.5. System requirements

Note: Hardware requirements for trading exchanges depend upon trading volume and the number of simultaneously connected trading clients.

Component	Minimum	Recommended
Processor	2 GHz+ Intel Pentium 4	Intel Xeon E7
RAM	12 GB low latency RAM	64+ GB low latency RAM
Hard drives	500 GB SATA 2	1 TB 15k RPM RAID 0 or SATA 3
Ethernet Adapter	1Gb Adapter	Intel X520 Dual 10Gb DA/SFP+

We recommend that you use the following BIOS settings:

- Disable Plug and Play OS.
- Set any parallel ports and onboard serial ports to specific addresses, if possible, instead of using an automatic setting.
- Enable SATA if your system includes a SATA drive.
- If your BIOS has an option for booting from "Other USB devices", disable it.

1.6. Before you start

Before you install this software, please note the following:

- NDAXCore requires Microsoft .NET Framework 4.5 or above. Although NDAXCore may run in Mono, it is not recommended.
  Please note that NDAXCore is intended for use by only accomplished,

1.7. Getting Started

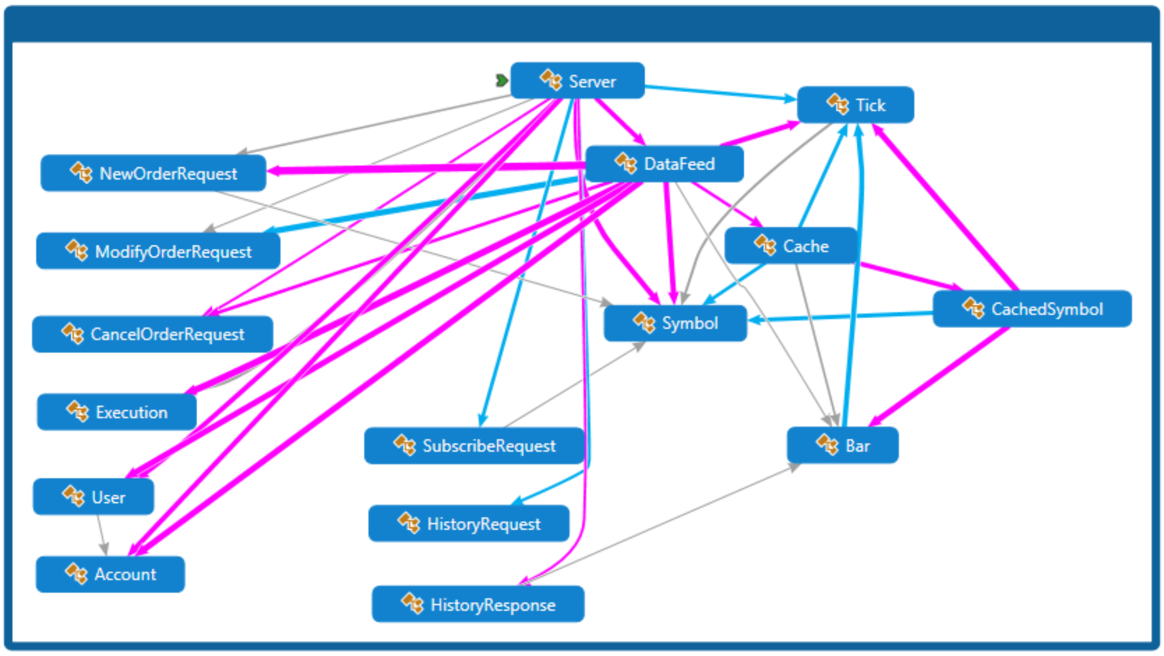
After you've installed NDAXCore, you'll find an extensive set of example code written in C#. You can run the NDAXCore server application (using Admin permissions) to explore the example projects and connect to the NDAXCore server, where you can begin entering simulated orders.

You can install and work with multiple versions of NDAXCore. Be sure that each NDAXCore instance runs on an independently unique port.

Classes and database structure 2

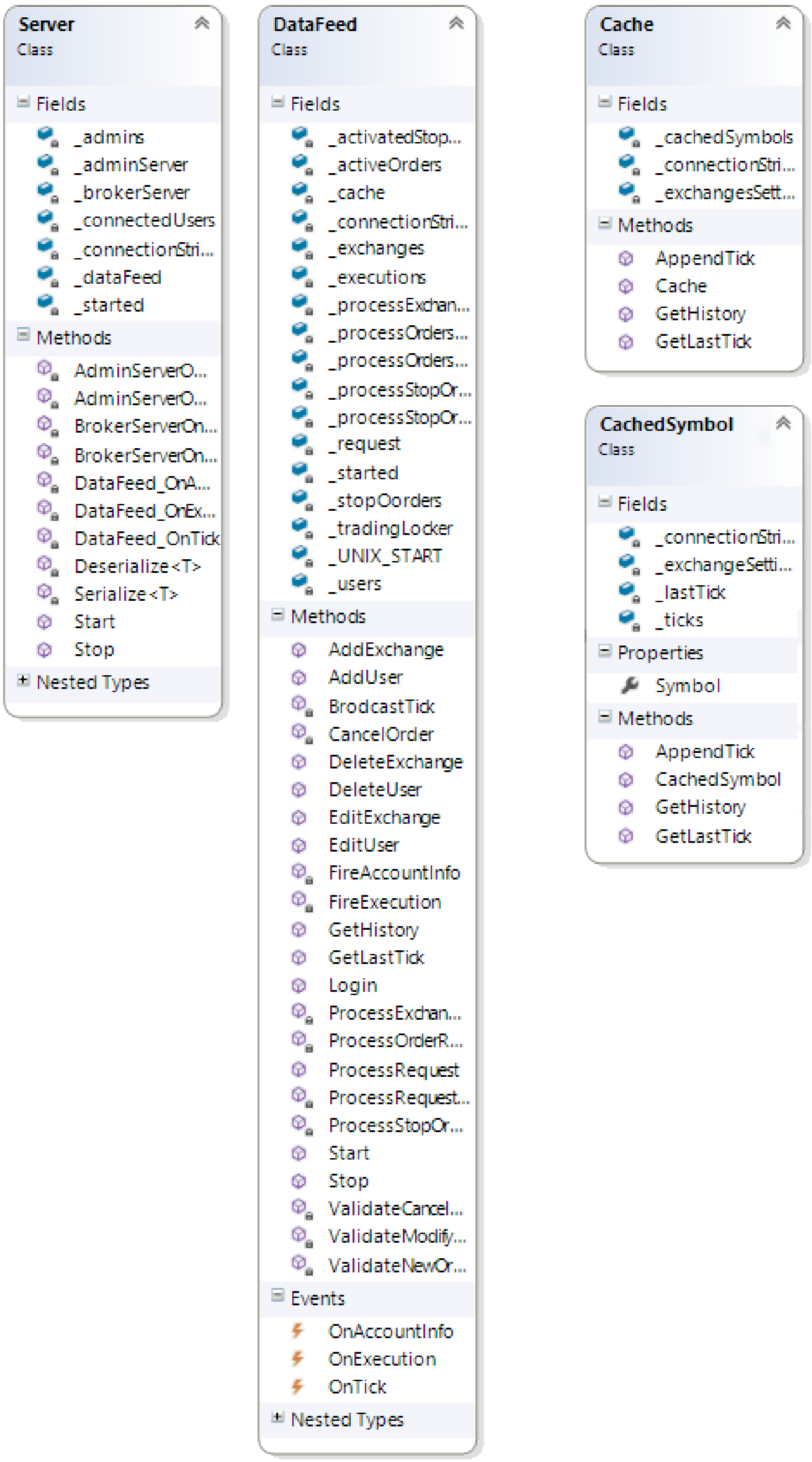
2.1. Class Relations

NDAXCore features the following class relations:



2.2. Class diagram

The following diagram outlines the class architecture of NDAXCore:



2.3. Class overview

Server — admin and trader requests processing class.
Public methods:

Name	Description
Start() : string	Initializes socket connections and starts requests processing
Stop() : void	Deinitializes socket connections and stops requests processing

DataFeed — order matching engine class.
Events:

Name	Description
OnTick(Action<Tick>)	Notification for new price
OnExecution(string, Execution)	Notification for new execution
OnAccountInfo(string, Account)	Notification for account balance modification
OnMessageResponse(string, string, string)	Notification for new message result

Private members:

Name	Description
_users : List<User>	List of users
_activeOrders : List<NewOrderRequest>	List of pending orders market and limit orders
_stopOorders : List<NewOrderRequest>	List of pending stop orders
_activatedStopOrders : Queue<NewOrderRequest>	Queue of activated by current execution stop orders
_executions : List<Execution>	List of executions
_tradingLocker : List<Execution>	Synchronisation object for multi threading
_requests : Queue<object>	Queue of received from users requests
_started : bool	Indicates engine status

_processOrdersEvent : ManualResetEvent	System event to process market and limit orders
_processStopOrdersEvent : ManualResetEvent	System event to process activated stop orders
_processOrdersThread : Thread	Thread to process market and limit orders
_processStopOrdersThread : Thread	Thread to process activated stop orders
_processExchangeSessionThread : Thread	Thread to process end of current day trading session
_connectionString : string	Database connection string
_exchanges : List<ExchangeSettings>	List of exchanges
_currencies : List<Currency>	List of currencies
_cache : Cache	Data storing engine
_unixStart : DateTime	Initial time for orders duration processing

Public methods:

Name	Description
Start() : void	Initialize exchange engine
Stop() : void	Uninitialize exchange engine
Login(LoginRequest) : UserLoginResponse	Validates user credentials
AddUser(User) : void	Adds user to users collection
EditUser(string, User) : void	Edits user in users collection
DeleteUser(string) : void	Deletes user in users collection
AddExchange(ExchangeSettings) : void	Adds exchange to exchanges collection
EditExchange(string, ExchangeSettings) : void	Edits exchange in exchanges collection
DeleteExchange(string) : void	Deletes exchange in exchanges collection
SetCurrencies(List<Currency>) : void	Sets available currencies
ProcessRequest(object) : void	Process user request
GetLastTick(Symbol) : Tick	Returns last price of the symbol
GetHistory(HistoryParameters) : List<Bar>	Process history data request

Private methods:

Name	Description
GetCurrencyMultiplier(string) : decimal	Returns currency multiplier
ProcessRequestHandler() : void	Process users requests
CancelOrder(CancelOrderRequest, bool) : void	Process order cancellation request
ValidateNewOrderRequest(NewOrderRequest) : string	Validates new order request
ValidateModifyOrderRequest (ModifyOrderRequest) : string	Validates order modification request
ValidateCancelOrderRequest (CancelOrderRequest) : string	Validates order cancellation request
ProcessOrderRequest(NewOrderRequest) : void	Processes order request according to matching rules
ProcessStopOrdersHandler() : void	Processes activated by last execution stop orders
ProcessExchangeSession() : bool	Compares current time to exchange session end time
FireExecution(string, Execution) : void	Generates notification of execution
FireAccountInfo(Account) : void	Generates notification of account balance modification
BrodcastTick(Tick, bool) : void	Generates notification of new price

Cache — market data storing engine class.
Private members:

Name	Description
_exchangesSettings : List<ExchangeSettings>	List of exchanges
_cachedSymbols : List<CachedSymbol>	List of symbols
_connectionString : string	Database connection string

Public methods:

Name	Description
AppendTick(bool) : void	Process new tick
GetHistory(HistoryParameters) : List<Bar>	Returns historical data
GetLastTick(Symbol) : void	Returns last price of symbol
ProcessExchangeSession(string) : void	Compares current time to exchange session end time

CachedSymbol — symbol market data storing engine class.
Public members:

Name	Description
Symbol : ExchangeSymbol	Symbol

Private members:

Name	Description
_ticks : List<Tick>	List of ticks
_tick : Tick	Last cached tick
_exchangeSettings : ExchangeSettings	Exchange settings
_connectionString : string	Database connection string

Public methods:

Name	Description
AppendTick(bool) : void	Process new tick
GetHistory(HistoryParameters) : List<Bar>	Returns historical data
GetLastTick() : Tick	Returns last price of symbol
ProcessExchangeSession() : void	Compares current time to exchange session end time

AdminSession — class for connected admins.
Public members:

Name	Description
UserName : string	User name
Session : WebSocketSession	Callback session

UserSession — class for connected users

Public members:

Name	Description
UserName : string	User name
Subscribers : List<SusbcibedItem>	Subscribed symbols
Session : WebSocketSession	Callback session

SubscribedItem — class for user subscribed symbols.

Public members:

Name	Description
Symbol : Symbol	Subscribed symbol
Currency : string	Subscribed symbol currency

Log — application log class.

Public methods:

Name	Description
WriteApplicationException(string, string, int, Exception) : void	Writes exception to log file
WriteApplicationInfo(string) : void	Writes info to log file
WriteApplicationExecution(NewOrderRequest, NewOrderRequest, Execution, Execution) : void	Writes execution details to log file
SendEmail(string) : void	Sends email

Message — base class for messages exchage.

Public members:

Name	Description
MessageType : string	Type of message
MessageID : string	ID of message

ExchangeSettings — settings of exchange class.

Public members:

Name	Description
Name : string	Exchange name
StartTime : TimeSpan	Exchange start time sessions

EndTime : TimeSpan	Exchange end time
Symbols : List<Symbol>	List of symbols
CommonCurrency : bool	Used for bitcoins exchange

LoginRequest — user login request class.

Public members:

Name	Description
UserName : string	User name
Password : string	Pasword

UserLoginResponse — user login response class.

Public members:

Name	Description
LoginResult : LoginResult	Login result
Symbols : List<Symbol>	List of exchange symbols
Currencies : List<string>	List of available currencies
Accounts : List<Account>	List of user accounts
Orders : List<NewOrderRequest>	List of user orders
Execuutions : List<Execution>	List of user's executions

AdminLoginResponse — admin login response class.

Public members:

Name	Description
LoginResult : LoginResult	Login result
Exchanges : List<ExchangeSettings>	List of exchangs
Currencies : List<Currency>	List of available currencies
Users : List<User>	List of users

LogoutRequest — user/admin logout request class.

Public members:

Name	Description
UserName : string	UserName

ServerInfoMessage — server sent user/admin info/error class – server forced user/admin disconnection class.

Public members:

Name	Description
Info : string	Message description reason
Exit : bool	Should user be disconnected

Note: Errors may occur due to inadequate computing resources, bugs in your code or network related issues. Examine the error string for information pertaining to the cause of the error.

Symbol — trade symbol class.

Public members:

Name	Description
Name : string	Symbol name
Exchange : string	Symbol exchange
Currency : string	Symbol currency

Currency — currency info class.

Public members:

Name	Description
Name : string	Currency name
Multiplier : decimal	Currency multiplier

User — user credentials data class.

Public members:

Name	Description
UserName : string	User name
Password : string	Pasword
Accounts : List<Account>	User accounts

Account — user account data class.

Public members:

Name	Description
Name : string	Accountname
Balance : decimal	Account balance
Currency : string	Account currency

SubscribeRequest — market data subscription request class.

Public members:

Name	Description
UserName : string	User name
Symbol : Symbol	Symbol to subscribe
Currency : string	Symbol currency to display
Subscribe : bool	Indicates subscribe or unsubscribe

Tick — real time tick data class.

Public members:

Name	Description
Symbol : Symbol	Tick symbol
Currency : string	Tick currency
DateTime : Time	Tick trade time
Bid : decimal	Tick bid price
BidSize : long	Tick bid size
Ask : decimal	Tick ask price
AskSize : long	Tick ask size
Price : decimal	Tick traded price
Volume : long	Tick volume

HistoryRequest — historical data request class.

Public members:

Name	Description
UserName : string	User name
Parameters : HistoryParameters	Parameters

HistoryParameters — historical data request parameters class.
Public members:

Name	Description
Symbol : Symbol	Symbol
Currency : string	Request currency to display
Periodicity : Periodicity	Periodicity
Interval : int	Interval
BarsCount : int	Number of bars

Bar — Bar data class.
Public members:

Name	Description
Time : DateTime	Bar trade time
Open : decimal	Bar open price
High : decimal	Bar high price
Low : decimal	Bar low price
Close : decimal	Bar close price
Volume : long	Bar volume

HistoryResponse — historical data response class.
Public members:

Name	Description
Bars : List<Bar>	Historical data

NewOrderRequest — new order request class – new order request class.
Public members:

Name	Description
ID : string	User defined order ID
Symbol : Symbol	Symbol
Time : DateTime	Sending time
ActivationTime : DateTime	Activation time of stop and stop limit orders
Account : string	User account
Side : Side	Order side
OrderType : OrderType	Order type
LimitPrice : decimal	Limit price

StopPrice : decimal	Stop price
Quantity : long	Order quantity
TimeInForce : TimeInForce	TimeInForce
ExpirationDate : DateTime	ExpirationDate

ModifyOrderRequest — order modification request class.

Public members:

Name	Description
ID : string	User defined order ID
Quantity : long	Order quantity
OrderType : OrderType	Order type
LimitPrice : decimal	Limit price
StopPrice : decimal	Stop price
TimeInForce : TimeInForce	TimeInForce
ExpirationDate : DateTime	ExpirationDate

CancelOrderRequest — order cancellation request class.

Public members:

Name	Description
ID : string	User defined order ID

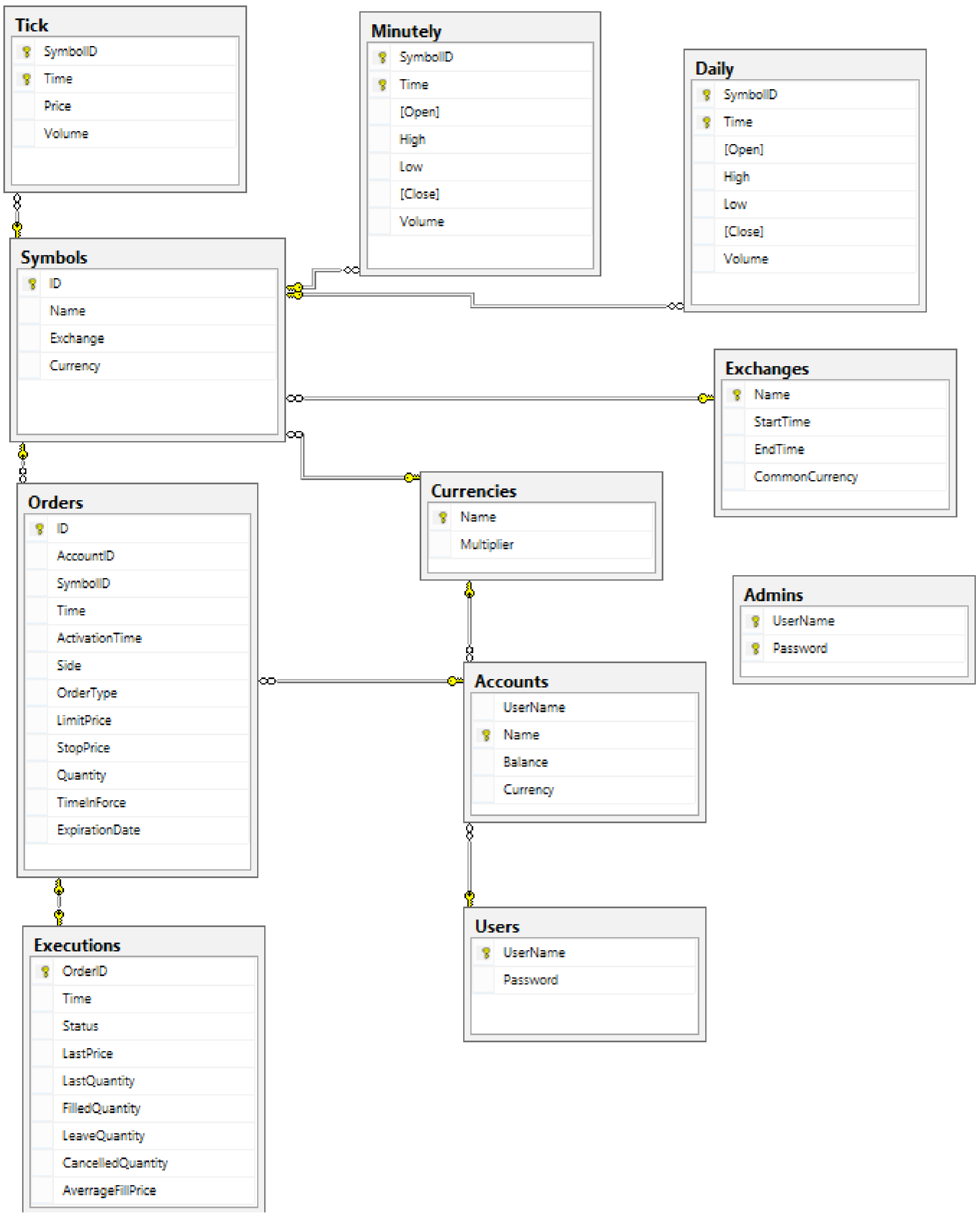
Execution — order executon report class.

Public members:

Name	Description
OrderID : String	Executed order id
Time : DateTime	Sending time
Status : status	Order status
LastPrice : decimal	Last filled price
LastQuantity : long	Last filled quantity
FilledQuantity : long	Total filled quantity
LeaveQuantity : long	Leave quantity
CancelledQuantity : long	Cancelled by user or system quantity
AverrageFillPrice : decimal	Averrage fill price
Message : string	Broker comment

2.4. Database structure

The following diagram outlines the NDAXCore database structure. Microsoft SQL Server is supported by default, which may be replaced with MySQL, Oracle or any other supported RDBMS.



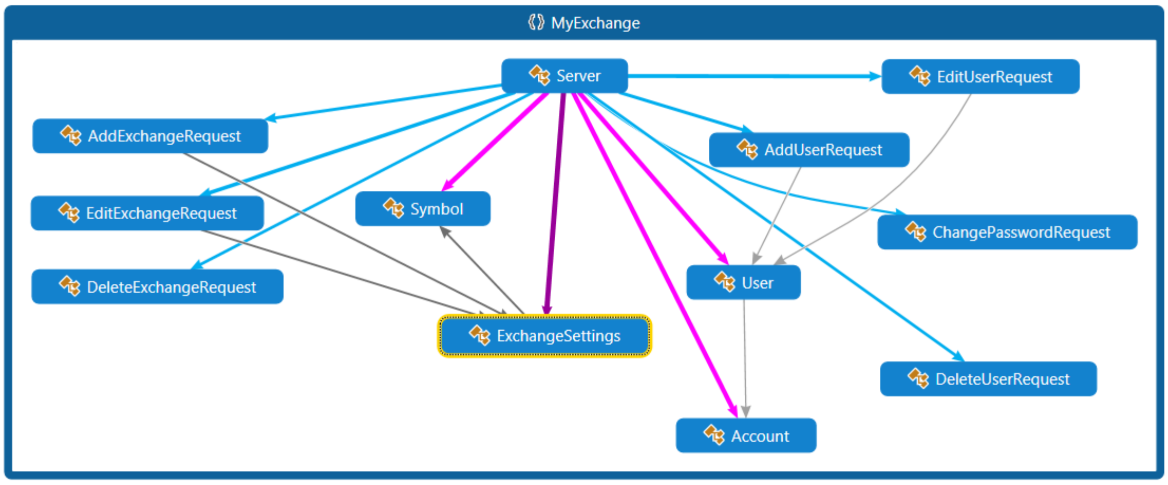
The database conains the following tables:

- **Admins** — exchange administrators;
- **Users** — user credentials;
- **Accounts** — user accounts;
- **Currencies** — available currencies;
- **Exchanges** — list of exchanges;
- **Symbols** — available for trading symbols;
- **Ticks** — ticks historical data;
- **Minutely** — minutely historical data;
- **Daily** — daily historical data;
- **Orders** — placed user orders;
- **Executions** — order executions.

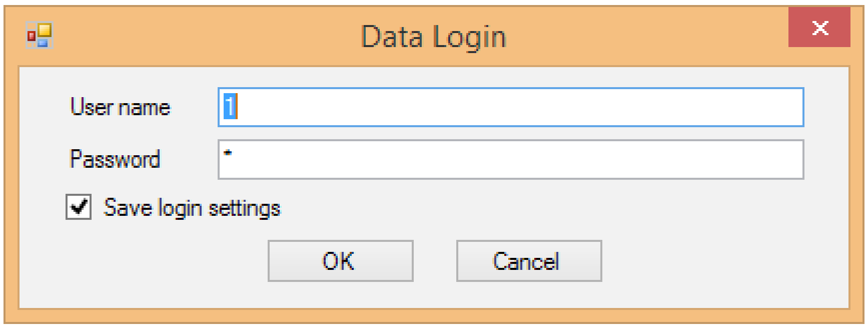
Administrator and user applications 3

3.1. Administrator application

The NDAXCore admin application allows the exchange operator to administrate the system and manage users, accounts, exchanges and symbols to some degree. Additional functionality can be added to the application. The following diagram illustrates the admin app architecture.

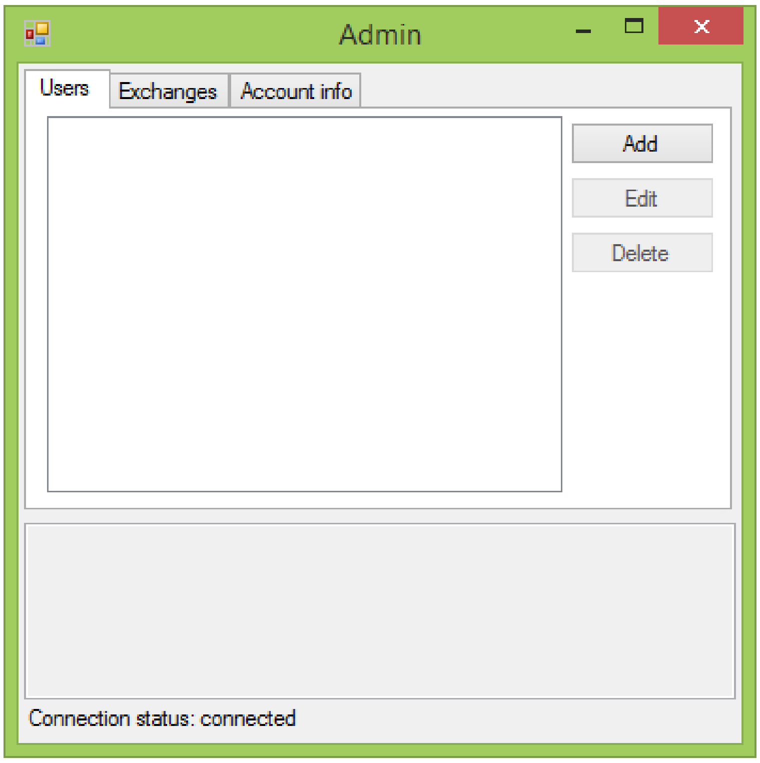


The following screenshots are presented as a walk-through of the administrator application:



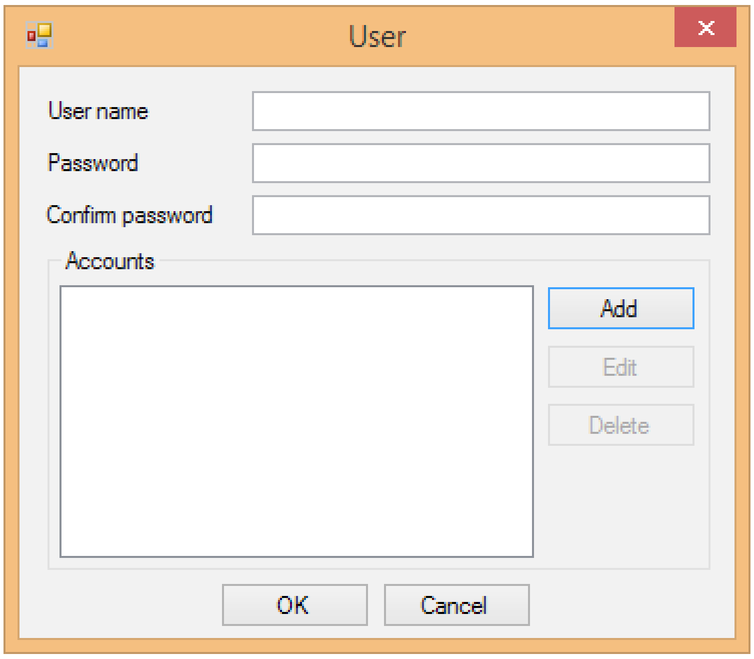
User Name — the admin user name stored in the database Admins table.
Password — the admin password stored in the database Admins table, in encrypted format.

Main window Users Tab (contains the list af all registered users):



Connection status: connected

Admin form to add new user (trader):



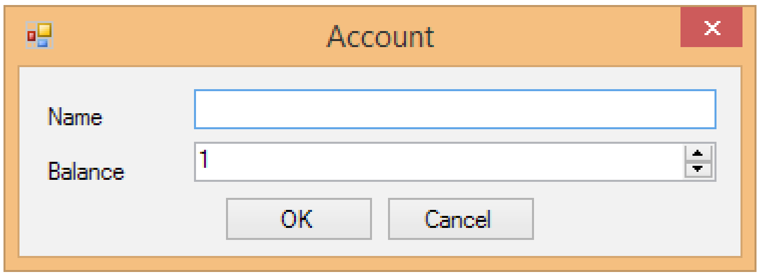
User name

Password

Confirm password

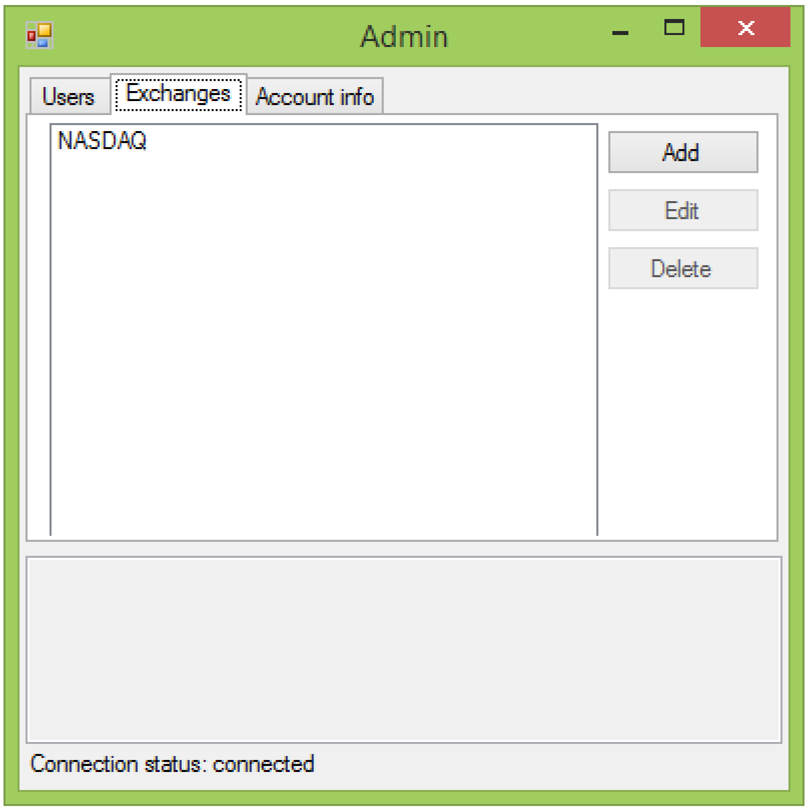
User Name — user names stored in the database Users table.
Password — user password stored in the database Users table, in encrypted format.
Accounts — a list of user trading accounts.

Form to add a new user account:



Name — the account name, stored in the database Accounts table.
Balance — the account balance, stored in the database Accounts table.

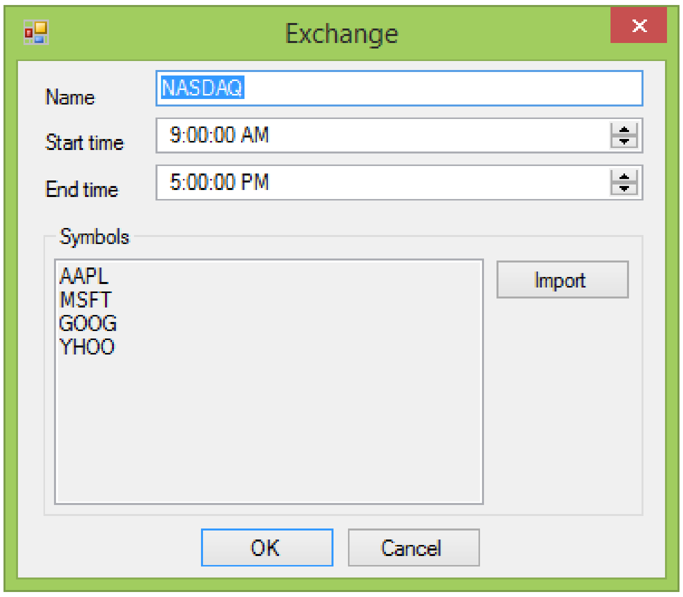
Main window Exchanges tab (contains a list of available exchanges):



NASDAQ

Connection status: connected

Form to add a new exchange:

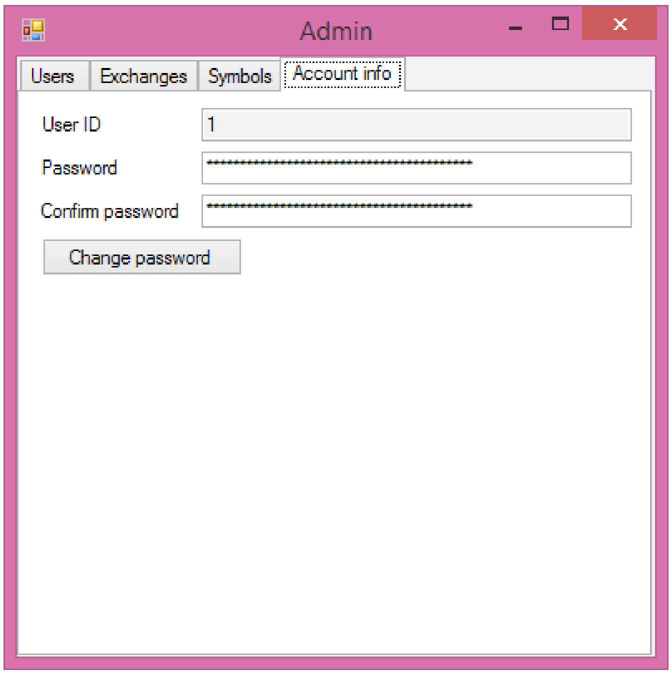


Name	NASDAQ
Start time	9:00:00 AM
End time	5:00:00 PM

AAPL
MSFT
GOOG
YHOO

Name — the exchange name, stored in the database Exchanges table.
Start time — exchange start time, stored in database Exchanges table.
End time — exchange table.
Symbols — exchange symbols, stored in database Symbols table. The admin is able to import symbols from a file. The supported file format is CSV.

Main window Admin account info tab (allows the admin to modify credentials):

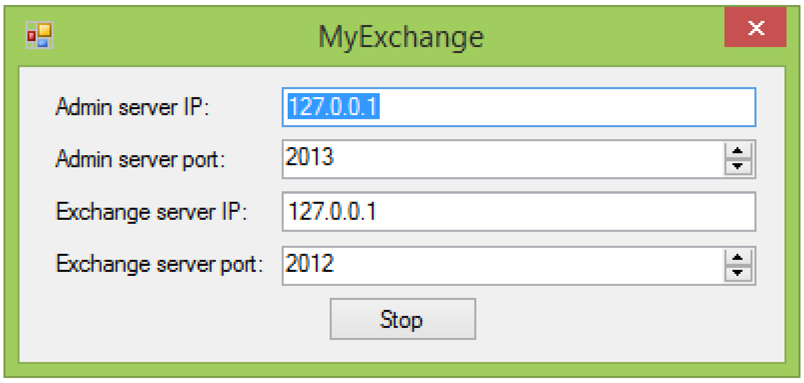


User ID	1
Password	**
Confirm password	**

3.2. Server installation

Installation instructions:

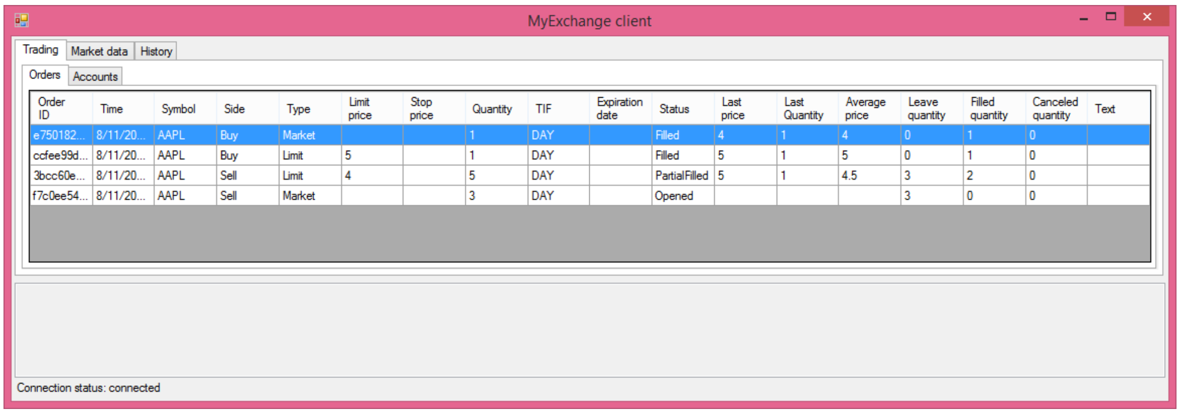
- Install the database. Run the provided SQL script from the DataBase.sql file in MS SQL Server Management studio. A new database will be created.
- Edit database connection string in NDAXCore.exe.config file. The file contains database connection parameters.
- Start the server application. Enter the IP addresses and ports of Admin server and Exchange server in the text fields of the server main window. Use the IP address on which the server will be hosted, and the associated port should be opened in the Windows Firewall and in your server's hardware firewall. For example, if you are using Amazon EC2, ensure that your Amazon Security settings allow the port to be open.



Admin server IP: 127.0.0.1
Admin server port: 2013
Exchange server IP: 127.0.0.1
Exchange server port: 2012

3.3. Example client application

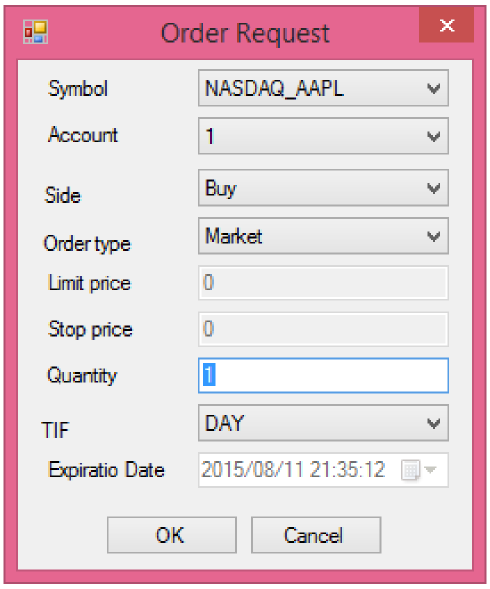
A C# and JavaScript client application are provided for example purposes.
Client application main window Trading tab:



Order ID	Time	Symbol	Side	Type	Limit price	Stop price	Quantity	TIF	Expiration date	Status	Last price	Last Quantity	Average price	Leave quantity	Filled quantity	Canceled quantity	Text
e750182...	8/11/20...	AAPL	Buy	Market			1	DAY		Filled	4	1	4	0	1	0	
ccfee99d...	8/11/20...	AAPL	Buy	Limit	5		1	DAY		Filled	5	1	5	0	1	0	
3bcc60e...	8/11/20...	AAPL	Sell	Limit	4		5	DAY		PartialFilled	5	1	4.5	3	2	0	
f7c0ee54...	8/11/20...	AAPL	Sell	Market			3	DAY		Opened				3	0	0	

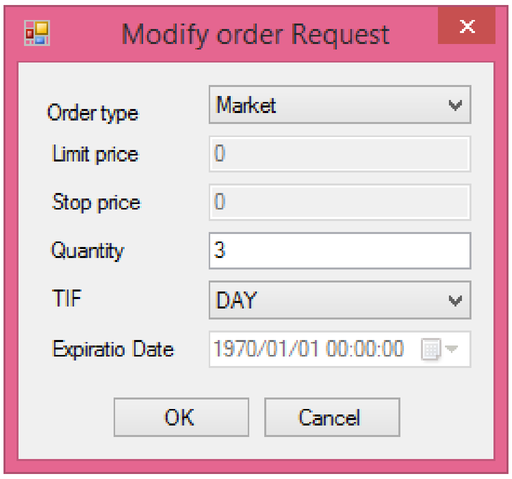
Connection status: connected

Client application new order window:



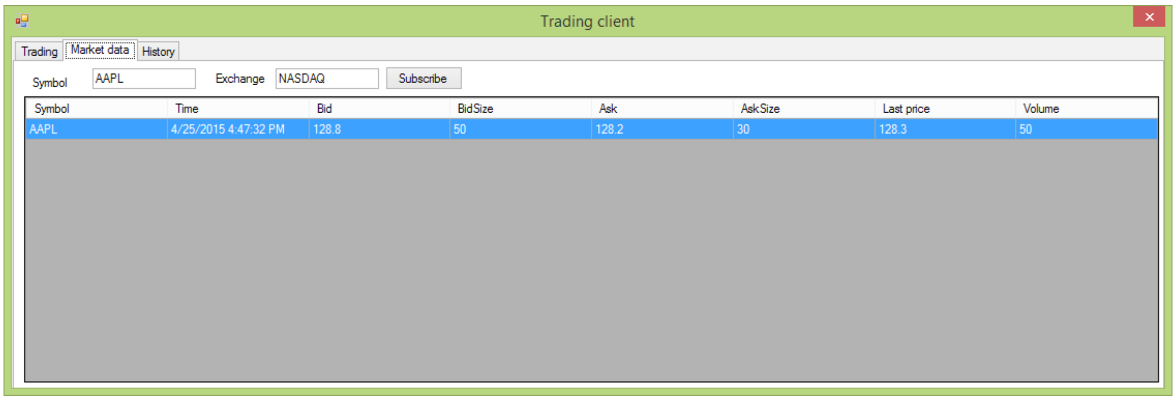
Symbol	NASDAQ_AAPL
Account	1
Side	Buy
Order type	Market
Limit price	0
Stop price	0
Quantity	1
TIF	DAY
Expiratio Date	2015/08/11 21:35:12

Client application order modification window:



Order type	Market
Limit price	0
Stop price	0
Quantity	3
TIF	DAY
Expiratio Date	1970/01/01 00:00:00

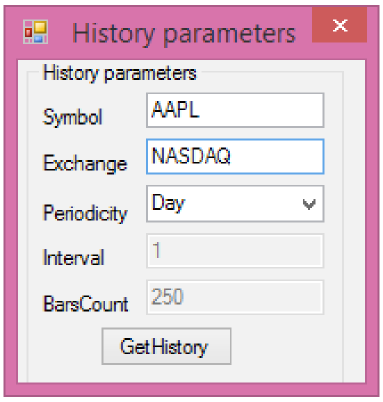
Client application main window Market data tab:



Symbol AAPL Exchange NASDAQ

Symbol	Time	Bid	BidSize	Ask	AskSize	Last price	Volume
AAPL	4/25/2015 4:47:32 PM	128.8	50	128.2	30	128.3	50

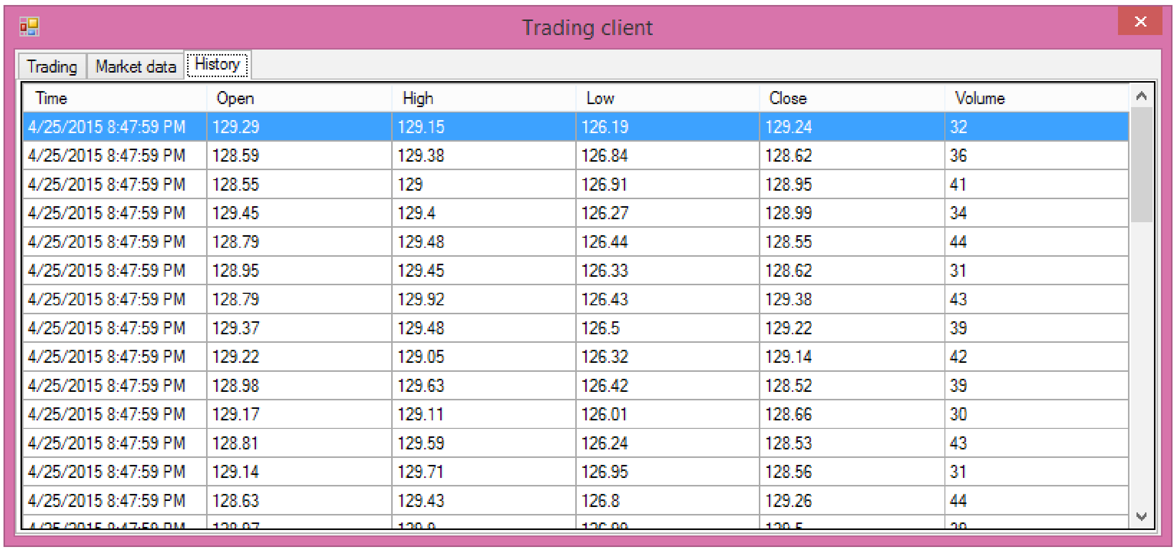
Client application historical data request parameters:



History parameters

Symbol	AAPL
Exchange	NASDAQ
Periodicity	Day
Interval	1
BarsCount	250

Client application main window history tab:



Time	Open	High	Low	Close	Volume
4/25/2015 8:47:59 PM	129.29	129.15	126.19	129.24	32
4/25/2015 8:47:59 PM	128.59	129.38	126.84	128.62	36
4/25/2015 8:47:59 PM	128.55	129	126.91	128.95	41
4/25/2015 8:47:59 PM	129.45	129.4	126.27	128.99	34
4/25/2015 8:47:59 PM	128.79	129.48	126.44	128.55	44
4/25/2015 8:47:59 PM	128.95	129.45	126.33	128.62	31
4/25/2015 8:47:59 PM	128.79	129.92	126.43	129.38	43
4/25/2015 8:47:59 PM	129.37	129.48	126.5	129.22	39
4/25/2015 8:47:59 PM	129.22	129.05	126.32	129.14	42
4/25/2015 8:47:59 PM	128.98	129.63	126.42	128.52	39
4/25/2015 8:47:59 PM	129.17	129.11	126.01	128.66	30
4/25/2015 8:47:59 PM	128.81	129.59	126.24	128.53	43
4/25/2015 8:47:59 PM	129.14	129.71	126.95	128.56	31
4/25/2015 8:47:59 PM	128.63	129.43	126.8	129.26	44

Subject Index

This chapter helps you to find information about the features, pages, windows, and other elements of NDAXCore. All the items are sorted alphabetically.

Note: Click the page number of the item and you will be navigated to the page with information about this element.