

NIAPI для роботов и приводов

Автор: Константин Ивайловский

В последнее время на рынке появляется все больше желающих сохранить нервы. Именно такие люди предоставляют своему роботу право биться за них на биржевой арене. Библиотека NIAPI, рассчитанная на игроков самого разного уровня, позволяет наилучшим образом автоматизировать торговлю. Она способна выставлять заявки в торговую систему с высокой скоростью и возможностью обратной связи.

Рынок похож на большую красно-зеленую арену, где не на шутку сражаются трейдеры: миллионеры, скальперы, арбитражеры, опционщики – люди или киборги и роботы. Обреченные не умирают, но теряют миллионы... нервных клеток. Все, кроме роботов...

Развитие торговых роботов

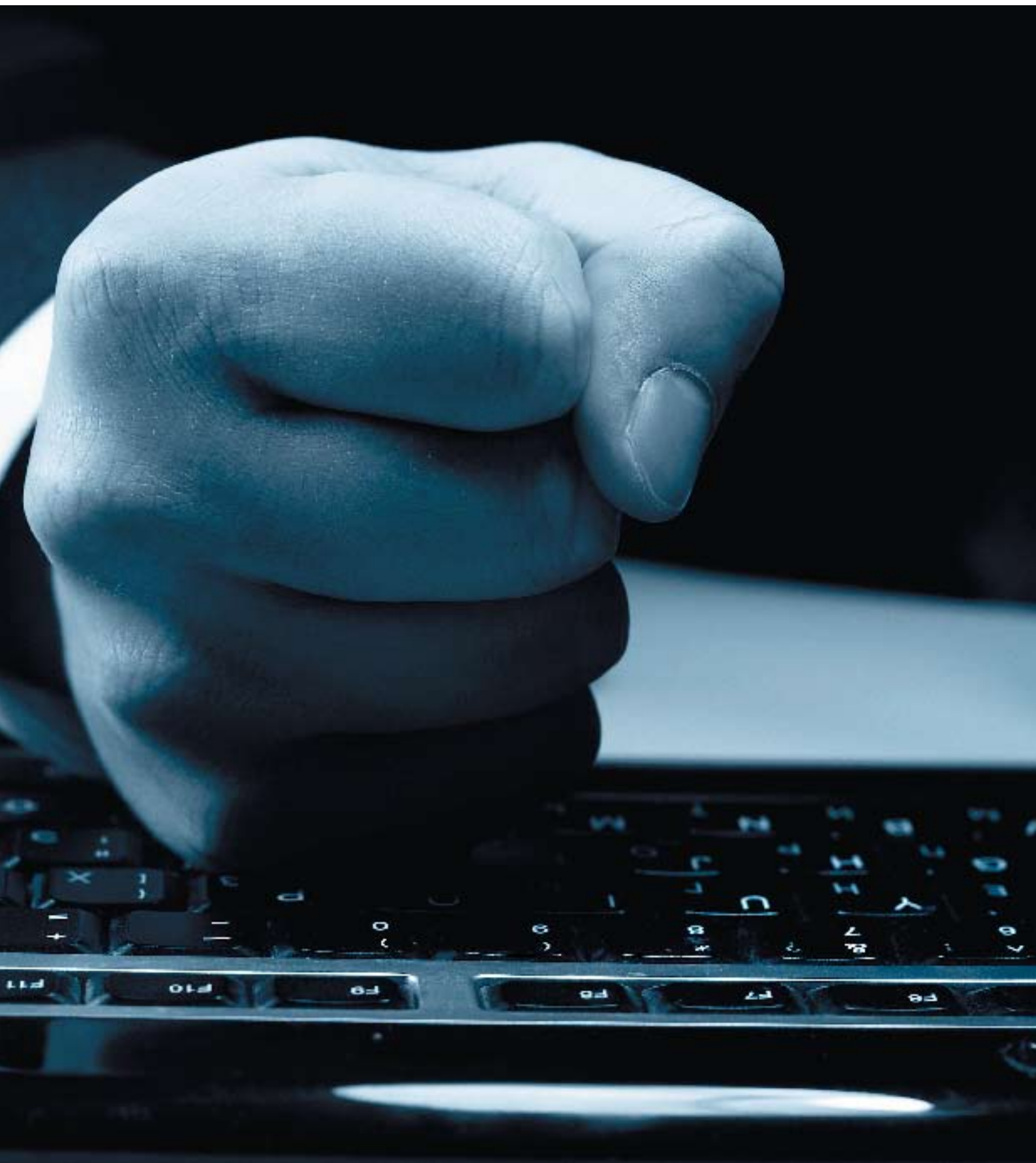
Механические торговые системы (МТС) с ручным вводом заявок. В первую очередь появились простые системы, которые из пакетов технического анализа подают сигналы трейдеру, передающему руками операции на биржу. Несмотря на огромный аналитический потенциал таких программ, как MetaStock или Omega Research, разработанные в их среде МТС с ручным вводом заявок самые медленные.

Передача данных от МТС к терминалу. Второй этап эволюции МТС – разработка программ и скриптов, которые передают

сформированные заявки в терминал с помощью текстовых файлов. «Подводные камни» такой связи усложняют работу робота. Во-первых, необходимо постоянное наблюдение за выставлением заявок, во-вторых, расчет количества инструментов возлагается на инвестора, в-третьих, нет проверки на лимиты, в-четвертых, есть задержка при считывании файла, в-пятых, при работе с несколькими инструментами сложность и задержки многократно увеличиваются. Используются и другие механизмы передачи данных, например, DDE пакета MS Office. Но все подобные решения объединяет такой огромный недостаток, как односторонний поток информации только от МТС к терминалу, что не позволяет следить за статусом отправленных заявок и сверяться с остатками портфелей.

Приложения, работающие через терминал. Такие решения адаптируются под конкретный программный терминал и мо-





Роботорговля

Рис. 1 Схема работы робота через терминал

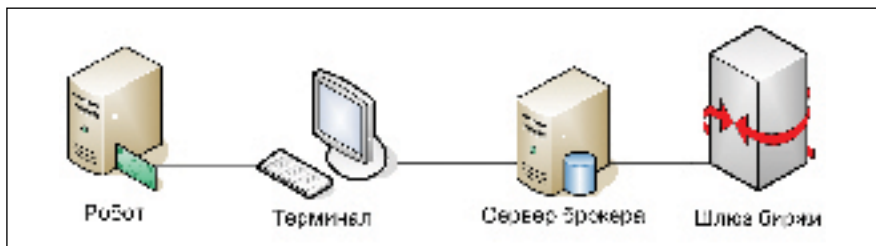


Рис. 2 Схема работы робота напрямую через шлюз биржи

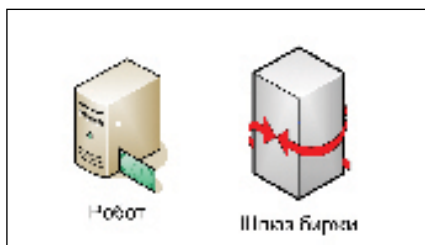
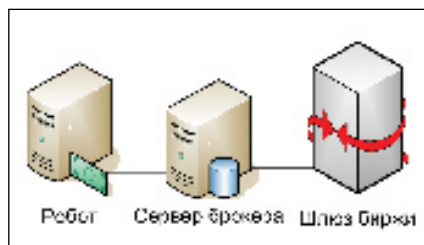


Рис. 3 Схема работы робота, взаимодействующего с сервером брокера через программный интерфейс



гут принимать вид отдельных приложений, модулей, дополнительных или встроенных скриптовых языков.

Работая через терминал, приложение-робот может получить всю необходимую информацию для самодиагностики, мониторинга счета и даже портфельного менеджмента. Примерами такой реализации являются приводы к терминалу Quik или скрипты QPILE, а также MetaTrader. Сверх того решаются задачи безопасности соединения и защиты финансовых операций. С другой стороны, простота оплачивается заметной медлительностью роботов и зависимостью от торговой платформы. Если терминал зависнет, то поток данных остановится, а вместе с ним прекратится и работа робота. Еще пара недостатков – задержка при обработке скриптов и передаче данных из терминала и, как правило, слабые возможности скриптов.

Автоматическая торговля через биржевой шлюз. Приложения, которые отправляют заявки напрямую в торговую систему биржи, минуя сервер брокера, ре-

агируют на сигналы быстрее всех остальных, но очень сложны с точки зрения реализации.

Что нужно трейдеру, чтобы подключить робота к шлюзу? Во-первых, необходимо иметь к нему доступ в обход брокерской системы контроля прав, лимитов и так далее. Во-вторых, знать протоколы биржевых шлюзов. В-третьих, постоянно следить за изменениями протоколов и стандартов биржи. Очевидно, что такие сложные решения по плечу профессиональным участникам рынка (брокерам, инвестиционным фондам, банкам), тогда как частным инвесторам они обходятся достаточно дорого.

Роботы, взаимодействующие с торговой системой брокера через API. Так что же в настоящий момент можно считать оптимальным решением для автоматической торговли? Если учесть все сильные и слабые стороны рассмотренных вариантов, можно предложить следующее: приложения-роботы взаимодействуют с сервером брокера через открытый программный интерфейс. Именно так складывается формула: мак-

симум быстродействия при минимуме усилий.

Использование открытого программного интерфейса к торговой платформе брокера дает разработчику следующие преимущества:

- реализация взаимодействия с биржей на уровне торговой платформы (без знания протоколов шлюзов);
- контроль прав и обеспечение безопасности на уровне торговой платформы;
- доступ к информации о портфелях и заявках;
- получение только необходимых данных;
- восстановление корректной работы после сбоев и обрыва связи;
- отсутствие задержек на обмен данными между роботом и торговой системой;
- минимальные задержки взаимодействия с биржевым шлюзом.

Такое полноценное API существует на западном рынке у системы CQG и системы NetInvestor на российском рынке. Рассмотрим для начала, как можно воспользоваться этими преимуществами, работая с системами технического анализа.

Роботы семейства Omega и MetaStock

Системы технического анализа (ТА) Omega и MetaStock содержат широкий набор возможностей для написания систем автоматической торговли, а также тестирования их на исторических данных (бэк-тестинг). Ими пользуются многие трейдеры, и каждый желает избежать задержек и ошибок при передаче сигнала на биржу через терминал. Наилучшим решением является отказ от терминала и отправка заявок сразу на сервер брокера. Интерфейс NIAPI является драйвером для осуществления быстрых операций между системами ТА и сервером брокера.

Omega TradeStation. Всего несколько строчек кода отличают

стратегию с заявками от стратегии с виртуальными сигналами. Однако прежде чем добавить заветные строки в алгоритм, необходимо загрузить библиотеку с сайта www.netinvestor.ru. Вся установка программы заключается в регистрации DLL в Windows и указании сервера и логина с паролем в приложении NiOmegaSrvr.exe.

Рассмотрим следующую стратегию с осциллятором RSI: когда индикатор пересекает линию перепроданности снизу, система покупает, когда же индикатор пересекает линию перекупленности сверху, система продает (см. Код 1).

Можно заметить, что с добавлением функций отправки приказов на сервер брокера код стратегии практически не изменился. Для этого мы всего лишь объявили функции и вместе с сигналами отправили заявки (см. Код 2).

Затем необходимо проверить исполнение заявок и обработать исключения неполного их исполнения. Дополнительно хочется отметить, что библиотека позволяет выставлять и снимать приказы, получать информацию об их исполнении, анализировать состояние портфеля. Перечисленные возможности обратной связи являются необходимым условием функционирования качественного робота.

Equis MetaStock. Аналогичная стратегия строится и в MetaStock, но для выставления заявки нужно написать индикатор, отправляющий ее на сервер, например: ExtFml(«NiOmegaDll.MSOrder», «S01-0000F00», «EQBR», «ЛКОH», «1111/», 1, 100). Для использования NIAPI в связке с MetaStock нужно скопировать NiOmegaDll.dll в директорию C:\Program Files\Equis\External function DLLs.

Для тестирования стратегий исторические данные принято брать с сайтов, таких как finam.ru или mfd.ru. Но роботам в системах ТА необходимо получать информацию о торгах в реальном

времени, поэтому для экспорта котировок соответствующим образом настраивают торговые терминалы Quik, NetInvestor и пр. Но в этом случае любые зависания терминала приведут к сбою в работе робота, так как поток рыночных данных остановится. Если бы не утилита TickTrack (www.ticktrack.ru), поставляющая рыночные данные прямо в Omega и MetaStock, то исключить торговый терминал как, вероятно, самый медленный элемент не удалось бы. А так, за счет сокращения цепочки при получении данных, робот быстрее реагирует на рыночные изменения. Избавление от необходимости «поддерживать» связь всех компонентов делает трейдинг немислимо простым.

NIAPI. Предназначение

Решение NetInvestor API предназначено для получения торговой информации для расширенной аналитики, а также выставления заявок с высокой скоростью и возможностью обратной связи. Осо-

бенностью NIAPI является наличие всех функций, используемых разработчиками сервера брокера.

Наиболее распространенные варианты использования библиотеки:

- **Расчеты в Microsoft Excel.** С помощью NIAPI в MS Excel можно поставлять данные непосредственно с сервера брокера. Такой способ получения информации избавляет от настройки всевозможных экспортов и увеличивает скорость поставки данных.

- **Аналитика во внешних программах технического анализа.** Системы технического анализа MetaStock или Omega TradeStation могут отправлять заявки непосредственно в торговую систему, минуя терминал, и получать информацию об их статусе. Робот в таком исполнении способен управлять позицией и реализовывать стратегии Money Management.

- **Автоматические торговые системы и роботы.** Высокочастотный трейдинг требует скорости в расчете алгоритмов и выставлении заявок. Используя NIAPI, можно

КОД 1

```
Inputs: RSILength(10), OverSold(30), OverBuy(70);
```

```
Variables: Seccode("RTS-12.09"), Secboard("PSFU"), Account("AC1"),
Brokerref("778"), Timeout(10000);
Var: result(7), output_conn(-7);
```

```
defineDLLFunc: "D:\NiOmega\niomegadll.dll", int, "conn",int;
defineDLLFunc: "D:\NiOmega\niomegadll.dll", int, "futaddorder",lpstr,lpstr,int,doubl
e,lpstr,lpstr,lpstr,lpstr,int;
defineDLLFunc: "D:\NiOmega\niomegadll.dll", int, "is_conn";
```

```
result = is_conn();
if result=0 then
    output_conn = conn(timeout);
```

```
If Currentbar > 1 AND RSI(Close, RSILength) Cross Over OverSold Then
begin
    Buy ("RSI-BUY") 1 contract on close Limit;
    futaddorder(seccode, "B", 1, Close, "L", brokerref, secboard, account, timeout);
end;
```

```
If Currentbar > 1 AND RSI(Close, RSILength) Cross Below OverBuy Then
begin
    Sell ("RSI-SELL") 1 contract on close Limit;
    futaddorder(seccode, "S", 1, Close, "L", brokerref, secboard, account, timeout);
end;
```

КОД 2

```
defineDLLFunc: "D:\NIOmega\niomegadll.dll", int, "conn",int;
defineDLLFunc: "D:\NIOmega\niomegadll.dll", int, "futaddorder",lpstr,lpstr,int,doubl
e,lpstr,lpstr,lpstr,lpstr,int;
defineDLLFunc: "D:\NIOmega\niomegadll.dll", int, "is_conn";
...
futaddorder(secocode, "B", 1, Close, "L", brokerref, secboard, account, timeout);
futaddorder(secocode, "S", 1, Close, "L", brokerref, secboard, account, timeout);
```

Рис. 4 Технология работы через NIAPI

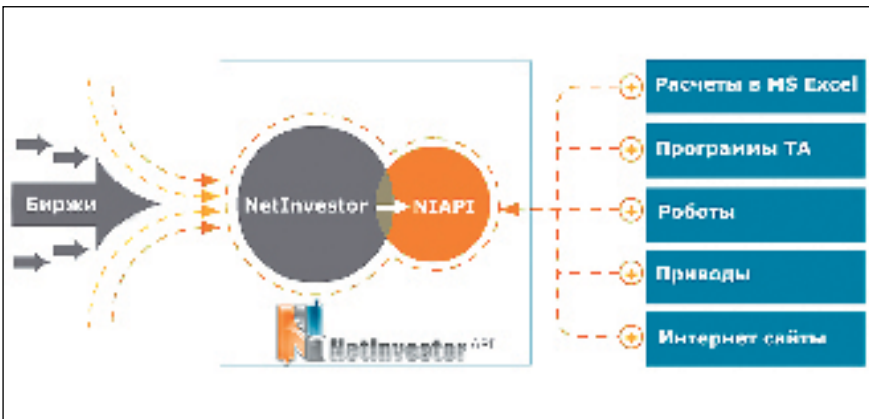
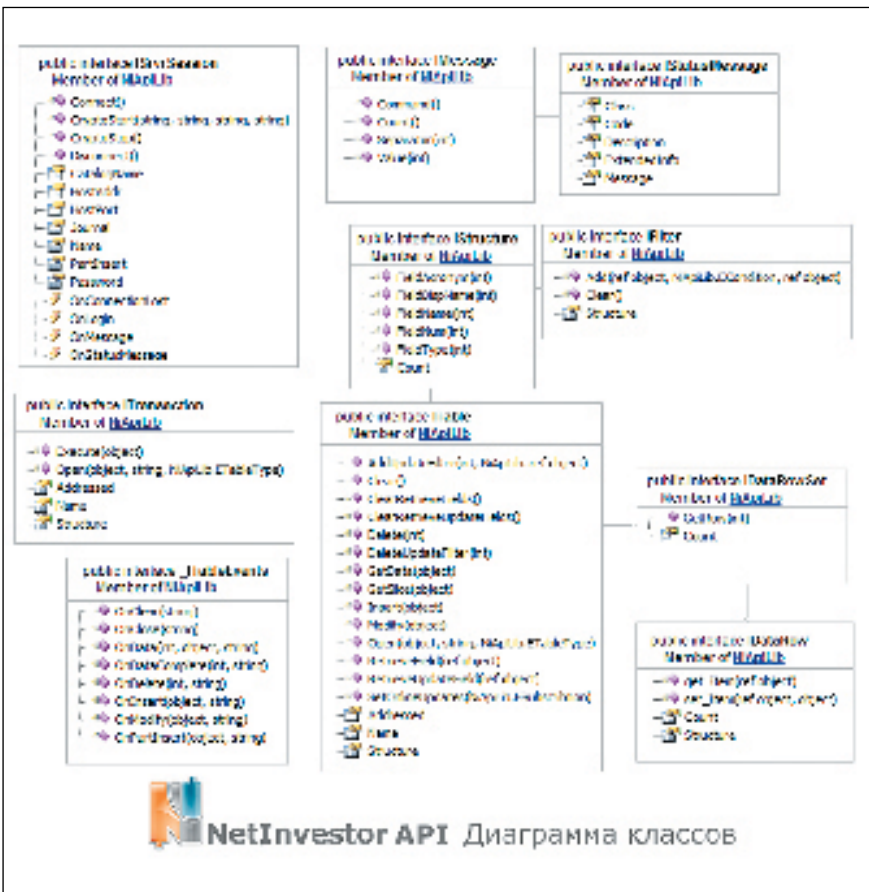


Рис. 5 Диаграмма классов NIAPI



написать оптимизированные системы, ограничивая получаемую из торговой системы информацию необходимым минимумом и быстрее производя вычисления. Цепочка компонентов при поставке информации в архитектуре робота минимальна.

• **Разработка сторонних приложений и приводов.** В последнее время значительно возрос спрос на «приводы» и «скальперские примочки». Вероятно, самый существенный недостаток таких приложений – зависимость от терминала, который замедляет реакцию на рыночные изменения за счет трансляции избыточной информации. NIAPI позволяет получать данные и отправлять заявки непосредственно на сервер брокера, обеспечивая полную независимость от торгового терминала.

• **Создание Internet-приложений.** Большинство брокерских компаний предоставляют Web-кабинет, позволяющий осуществлять анализ операций клиента и следить за изменениями цен финансовых инструментов. NIAPI может использоваться для связи последнего с системой интернет-трейдинга.

• **Интеграция с системами Back-Office.** Возможно применение для любых интеграций внутри инфраструктуры брокерских компаний.

Библиотека NetInvestor API осуществляет поставку рыночных данных с различных бирж в сторонние приложения, которым доступны и торговые функции. Информация текущей сессии транслируется в режиме реального времени, при этом есть возможность одновременно запросить исторические данные. Интерфейс NIAPI базируется на серверной технологии COM и поддерживает широкий набор языков программирования.

Рыночные данные, получаемые с помощью NIAPI:

- **Котировки** в реальном времени.
- **Все сделки.** Список всех сделок либо получение информации только о последней сделке.

- **Котировки второго порядка** (стаканы). Очередь заявок на покупку и продажу.
- **Новости.** Информационная поддержка и новостные ленты.
- **Счета.** Список счетов клиента.
- **Портфель.** Получение данных по позициям клиента.
- **Заявки.** Выставление всех видов заявок, получение информации о заявках.
- **Свои сделки.** Получение информации о своих сделках.
- **Стоп-лоссы и тэйк-профиты.** Выставление стоп-лоссов и тэйк-профитов, информации об их выставлении и результатах срабатывания.
- **Ценные бумаги.** Список и параметры торгуемых инструментов.

Пользователи могут также запросить данные по комиссиям, своим счетам и прочие сведения, доступные на сервере, который автоматически проверяет права доступа к различной информации.

Дополнительные преимущества, которыми пользуются брокерские, инвестиционные и управляющие компании:

Консолидированная информация с бирж. Библиотека NIAPI поставяет финансовую информацию по любым тикерам и с любой биржи, используя единый протокол в режиме реального времени, и позволяет реализовывать стратегии хеджирования или арбитража.

Новости и информационная поддержка. Информационная поддержка МФД-ИнфоЦентр включает новостные ленты ведущих агентств: РИА «Новости», «Интерфакс», ПРАЙМ-ТАСС, МФД-ИнфоЦентр, а также данные с мировых рынков по товарам, индексам и валютным парам.

NIAPI. Внутреннее строение

Архитектура NIAPI состоит всего из нескольких классов, позволяющих реализовать все возможности работы с сервером NetInvestor:

КОД 3

```
NiApiLib.SrvrSession srvrSession = new NiApiLib.SrvrSessionClass();
String catalogName = "C:\NIPRO\catalog.xml";
srvrSession.HostAddr = "nidemo.mfd.ru";
srvrSession.HostPort = "2900";
srvrSession.Name = "trader";
srvrSession.Password = "robot";
srvrSession.Connect();
...
srvrSession.Disconnect();
```

КОД 4

```
srvrSession.OnMessage += new _ISrvrSessionEvents_OnMessageEventHandler
(onMessage);
srvrSession.OnLogin += new _ISrvrSessionEvents_OnLoginEventHandler(onLogin);
srvrSession.OnConnectionLost += new _ISrvrSessionEvents_OnConnectionLostE
ventHandler(onConnectionLost);

public void onMessage(object Msg) {}
public void onLogin(int LoginResult){}
public void onConnectionLost({})
```

КОД 5

```
private NiApiLib.SrvrSession srvrSession;
private NiApiLib.Transaction niTransaction;

string I_SECCODE = "RTS-12.09"; //I_SECCODE
string I_BUYSELL = "S"; //I_BUYSELL
string I_QUANTITY = "1"; //I_QUANTITY
string I_PRICE = "145000"; //I_PRICE
string I_MKTLIMIT = "M"; //I_MKTLIMIT
string I_BROKERREF = "1"; //I_BROKERREF
string I_SECBOARD = "PSFU"; //I_SECBOARD
string I_ACCOUNT = "PSFU"; //I_ACCOUNT

Dictionary<int, string> dict = new Dictionary<int, string>();
dict.Add(2, I_SECCODE);
dict.Add(3, I_BUYSELL);
dict.Add(4, I_QUANTITY);
dict.Add(5, I_PRICE);
dict.Add(6, I_MKTLIMIT);
dict.Add(7, I_BROKERREF);
dict.Add(8, I_SECBOARD);
dict.Add(11, I_ACCOUNT);

niTransaction.Open(this.srvrSession, "FUTADDORDER", ETableType.tabAdrr);

NiApiLib.DataRow row = new NiApiLib.DataRow();
row.Structure = niTransaction.Structure;

object o = new object();
ICollection<int> c = dict.Keys;
foreach(int i in c) {
    o = i;
    row.set_Item(ref o, (object)dict[i]);
}
this.niTransaction.Execute((object)row);
```

КОД 6

```

System.Data.DataTable dtTickers;
NiApiLib.Table tbl;
NiApiLib.DataRowSet myDataRowSet;
NiApiLib.Structure myStructure;
NiApiLib.Filter myFiltr;

// Создаем обработчики
tbl = new TableClass();
tbl.OnData += new _ITableEvents_OnDataEventHandler(onData);
tbl.OnDataComplete += new _ITableEvents_OnDataCompleteEventHandler(onDataComplete);
tbl.OnInsert += new _ITableEvents_OnInsertEventHandler(onInsert);
//Открытие таблицы
myStructure = new NiApiLib.StructureClass();
myFiltr = new FilterClass();
tbl.Open(session.getNISrvrSession(), "ORDERS_PSFU", NiApiLib.ETableType.tabNoneAddr);
tbl.SetOnlineUpdates(NiApiLib.ESubscription.subEnabled);
myStructure=(NiApiLib.Structure)this.tbl.Structure;
myFiltr.Structure=this.tbl.Structure;
tbl.GetData(this.myFiltr);
...
Функции обработчиков
void onInsert(object niDataRow, string tableName){}
void onDataComplete(int Code, string tableName) {}
void onData(int Code, object DataRowSet, string TableName){}

```

КОД 7

```

private void onInsert(object niDataRow, string tableName)
{
    if (tableName.Equals("ORDERS_PSFU") || tableName.Equals("orders_psfu"))
    {
        NiApiLib.DataRow myrow = (NiApiLib.DataRow)niDataRow;
        object o = new object();
        //GET field by number
        int t = 0;
        o = (object)t;
        string orderId = myrow.get_Item(ref o).ToString();
    }
}

```

- **ISrvrSession** – реализация коммуникации и протокольной части, поддерживает работу через прокси службы, протокольное сжатие, бинарный протокол, передачу и прием защищенных сообщений (ЭЦП и шифрование);
- **IMessage** – интерфейс для работы с протокольными сообщениями;
- **IStatusMessage** – интерфейс для работы с протокольными статусными сообщениями и сообщениями об ошибках;

- **IStructure** – интерфейс для работы с описанием структур таблиц и форматов транзакций;
- **IDataRow** – интерфейс для доступа к «полям записи» таблиц и данным о транзакциях;
- **IDataRowSet** – интерфейс для доступа к коллекциям IDataRow;
- **IFilter** – интерфейс доступа к коллекциям фильтров для получения исторических данных по таблицам;
- **ITable** – интерфейс для работы с серверными таблицами, поддерживает описание струк-

туры таблиц, запросы истории и срезов, запросы на вставку, модификацию и удаление записей, управление подпиской на обновления;

- **ITransaction** – интерфейс для работы с серверными транзакциями, поддерживает описание структуры транзакции и запросы на их выполнение.

Рассмотрим основные функции, с которыми сталкивается разработчик роботов:

- Соединение с сервером;
- Выставление заявки;
- Получение информации о выставленной заявке.

Первым шагом является получение библиотеки NiApi.dll. Ее регистрация в ОС Windows осуществляется командой `regsvr32 NiApi.dll`.

Соединение с сервером NetInvestor. Для соединения с сервером необходимо знать IP-адрес, порт сервера, логин с паролем, а также иметь файл `catalog.xml`, хранящий информацию о таблицах сервера. Файл можно получить из директории торгового терминала `NetInvestor Professional` (см. Код 3).

Запрос на соединение с сервером отправлен, но необходимо обработать ответ от него о подключении и другие сообщения. Регистрация обработчиков очень проста (см. Код 4).

Реализация подключения к серверу на этом завершена. Для простоты начало сессии с сервером можно написать в отдельном классе `SrvrSession`.

Выставление заявки. Отправка заявки или выполнение любой другой операции осуществляется созданием объекта `ITransaction` с названием транзакции и передачей ей параметров. Операция исполнится в рамках сессии подключения к серверу, поэтому достаточно трех объектов: сессия, транзакция и строка со значениями. Список транзакций и их параметры можно найти в `catalog.xml` (см. Код 5).

Подписка на рыночные данные. Подписка на любые данные в NIAPI осуществляется с указанием таблицы, из которой эта информация получается. Например, таблица QUOTES содержит котировки, PORTFOLIO - данные о состоянии портфеля пользователя. В общем, работа с данными осуществляется схожим образом, и мы рассмотрим ее на примере получения информации о выставленной нами ранее заявке. Сведения о заявках доступны в таблице ORDERS_PSFU, описание которой мы находим в файле catalog.xml (см. Код 6).

Теперь выполним обработку функции onInsert. После выставления приказа в таблицу ORDERS_PSFU добавится (onInsert) новая запись об отправленной заявке (см. Код 7).

NIAPI позволяет применять классы как для написания своего каркаса приложения, так и использования текущей архитектуры в целях реализации конкретных задач.

Приводы с NIAPI

Открытый программный интерфейс NIAPI с успехом используется для разработки программного обеспечения, которое расширяет функционал торговой платформы NetInvestor. Помимо многочисленных роботов в качестве примера полезного и удачного использования NIAPI можно привести программу-привод SpecMachineNZ и TrinityNZ, разработанные «Инновационной компанией НЗ».

SpecMachineNZ – роботизированное рабочее место трейдера для торговли на рынке FORTS. Приложение работает у любого брокера, который использует торговую платформу NetInvestor. Оно включает в себя модуль «Спекулянт», предназначенный для построения стратегии и автоматической торговли фьючерсами, «ручной» модуль «Торговля спрэдом» и аналитический модуль «Доска опционов» с приказами-триггерами.

TrinityNZ – аналитическо-торговая программа, позволяющая реализовывать широкий набор

стратегий: хеджирования, арбитража, спекулятивные и торговли опционами.

Банки и брокерские компании, используя NIAPI, написали около сотни роботов, автоматизаций и собственных стратегий. Возможности NIAPI безграничны и содержат все, что нужно для торговли.

Итоги

Библиотека NIAPI позволяет автоматизировать работу пользователей любого уровня: от инвесторов, использующих возможности систем технического анализа Omega TradeStation, MetaStock, пакетов анализа данных MS Excel и MatLab, трейдеров-программистов, пишущих «приводы» и торговых роботов на языках высокого уровня, до профессиональных участников, которые автоматизируют как торговлю, так и работу Back-Office. Автоматизация с помощью NIAPI характеризуется широким набором возможностей, высокой скоростью работы и уровнем надежности.

F&O

