

Открытый интерфейс «SmartCOM»

Версия от 4-04-2010

Оглавление

1. УСТАНОВКА И УДАЛЕНИЕ	2
2. ВЗАИМОДЕЙСТВИЕ С ОТКРЫТЫМ ИНТЕРФЕЙСОМ	2
2.1. Подключение к своему приложению.....	2
3. СОБЫТИЯ	3
3.1. СОБЫТИЕ ADDBAR.....	3
3.2. СОБЫТИЕ ADDPORTFOLIO.....	3
3.3. СОБЫТИЕ ADDSYMBOL.....	4
3.4. СОБЫТИЕ ADDTICK.....	4
3.5. СОБЫТИЕ ADDTRADE.....	5
3.6. СОБЫТИЕ CONNECTED.....	5
3.7. СОБЫТИЕ DISCONNECTED.....	6
3.8. СОБЫТИЕ ORDERFAILED.....	6
3.9. СОБЫТИЕ ORDERSUCCEEDED.....	6
3.10. СОБЫТИЕ SETPORTFOLIO.....	7
3.11. СОБЫТИЕ UPDATEBIDASK.....	7
3.12. СОБЫТИЕ UPDATEORDER.....	7
3.13. СОБЫТИЕ UPDATEPOSITION.....	8
3.14. СОБЫТИЕ UPDATEQUOTE.....	9
4. МЕТОДЫ	10
4.1. МЕТОД CANCELALLORDERS.....	10
4.2. МЕТОД CANCELBIDASK.....	10
4.3. МЕТОД CANCELORDER.....	10
4.4. МЕТОД CANCELPORTFOLIO.....	10
4.5. МЕТОД CANCELQUOTES.....	11
4.6. МЕТОД CANCELTICKS.....	11
4.7. МЕТОД CONNECT.....	11
4.8. МЕТОД DISCONNECT.....	14
4.9. МЕТОД GETBARS.....	14
4.10. МЕТОД GETPRORTFOLIOLIST.....	17
4.11. МЕТОД GETSYMBOLS.....	18
4.12. МЕТОД ISCONNECTED.....	21
4.13. МЕТОД LISTENBIDASKS.....	22
4.14. МЕТОД LISTENPORTFOLIO.....	22
4.15. МЕТОД LISTENQUOTES.....	22
4.16. МЕТОД LISTENTICKS.....	22
4.17. МЕТОД MOVEORDER.....	23
4.18. МЕТОД PLACEORDER.....	23

1. Установка и удаление.

Установка:

Для установки интерфейса SmartCOM, скачайте и запустите мастер установки программного обеспечения, расположенный по адресу:

<http://www.itinvest.ru/software/download/>

Далее следуйте его указаниям.

Удаление:

Для удаления интерфейса SmartCOM, зайдите: Пуск → Панель управления → Установка и удаление программ. Выберите из списка SmartCOM 2. Нажмите Удалить, и следуйте указаниям мастера удаления.

2. Взаимодействие с открытым интерфейсом

COM-модуль «SmartCom» позволяет:

- Подавать заявки.
- Получать информацию о рынке.
- Подключаться/отключаться от сервера котировок ITinvest.

2.1. Подключение к своему приложению

В настоящее время, существует множество языков программирования позволяющих использовать COM технологии. Ниже будут рассмотрены варианты подключения открытого интерфейса к внешним приложениям, написанных на C#.

Для подключения COM модуля к проекту необходимо:

1. Нажать правой кнопкой мыши на References в Solution Explorer и выбрать пункт Add reference...
2. Переключиться на вкладку COM
3. Выбрать «**SmartTrade client library**» и нажать «Select»
4. После чего нажать ОК

Для использования в проекте необходимо подключить через **using** модуль **StClientLib**. Создание объекта выглядит следующим образом:

```
[C#]  
StServer SmartCOM = new StClientLib.StServer();
```

3. События

Важное преимущество SmartCOM, данные поступают событиями, вам достаточно подписаться на данные. Технология COM является не многопоточной, поэтому рекомендуется обработчики событий делать как можно более лёгкими, и использовать буферы. Также необходимо учитывать, что события приходят построчно.

3.1. Событие AddBar

Происходит при получении исторических данных по инструменту в интервале.

[C#]

```
void AddBar(string symbol, StClientLib.StBarInterval interval, System.DateTime datetime, double open, double high, double low, double close, double volume)
```

Наименование	Тип	Описание
Symbol	String	Код ЦБ из таблицы котировок SmartTrade
Interval	StBarInterval	Интервал времени. StBarInterval_1Min = 1, StBarInterval_5Min = 2, StBarInterval_10Min = 3, StBarInterval_15Min = 4, StBarInterval_30Min = 5, StBarInterval_60Min = 6, StBarInterval_2Hour = 7, StBarInterval_4Hour = 8, StBarInterval_Day = 9, StBarInterval_Week = 10, StBarInterval_Month = 11, StBarInterval_Quarter = 12, StBarInterval_Year = 13
Datetime	DateTime	Дата и время интервала
Open	Double	Цена первой сделки после открытия в интервале
High	Double	Максимальная цена сделки в интервале
Low	Double	Минимальная цена сделки в интервале
Close	Double	Цена последней сделки в интервале
Volume	Double	Объём сделок в интервале

3.2. Событие AddPortfolio

Происходит при обновлении списка счетов доступных по текущему аккаунту

[C#]

```
void AddPorfolio(int row, int nrows, string portfolioName, string portfolioExch)
```

Наименование	Тип	Описание
Row	int	Номер счета в списке
NRows	int	Всего счетов в списке
portfolioName	string	Наименование портфеля
portfolioExch	string	Площадка доступная для портфеля

3.3. Событие AddSymbol

Происходит при обновлении справочника.

[C#]

```
void AddSymbol(int row, int nrows, string symbol, string short_name, string long_name, string type, int decimals, int lot_size, double punkt, double step, string sec_ext_id, string sec_exch_name, System.DateTime expiryDate)
```

Наименование	Тип	Описание
Row	int	Номер инструмента в справочнике
NRows	int	Всего инструментов в справочнике
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Short_name	string	Краткое наименование
Long_name	string	Полное наименование
Type	string	Код типа из справочника SmartTrade
Decimals	int	Точность цены
Lot_size	int	Размер лота ценных бумаг
punkt	double	Цена минимального шага
step	double	Минимальный шаг цены
sec_ext_id	string	ISN
sec_exch_name	string	Наименование площадки
expiryDate	DateTime	Дата экспирации

3.4. Событие AddTick

Происходит при возникновении сделки на рынке по инструменту.

[C#]

```
void AddTick(string symbol, System.DateTime datetime, double price, double volume, string tradeno)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Datetime	DateTime	Время сделки
Price	double	Цена сделки
Volume	double	Объём сделки
TradeNo	string	Id сделки на рынке

3.5. Событие AddTrade

По счёту произошла сделка.

Примечание: Для получения этого события необходимо наблюдать за торговым счётом, метод *ListenPortfolio*.

[C#]

```
void AddTrade(string portfolio, string symbol, string orderNo, double price, double amount, System.DateTime datetime, string tradeno)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
OrderNo	string	Номер заявки на сервере котировок
Price	double	Цена сделки
AMount	double	Объём сделки, если положительный покупка, отрицательный в случае продажи
Datetime	DateTime	Время сделки
TradeNo	string	Id сделки на рынке

3.6. Событие Connected

Связь с сервером котировок установлена.

[C#]

```
void Connected()
```

3.7. Событие **Disconnected**

Связь с сервером котировок разорвана.

```
[C#]
```

```
void Disconnected(string reason)
```

Наименование	Тип	Описание
Reason	string	Причина

3.8. Событие **OrderFailed**

При выставлении заявки, произошла ошибка.

Примечание: Для получения этого события необходимо наблюдать за торговым счётом, метод [ListenPortfolio](#).

```
[C#]
```

```
void OrderFailed(int cookie, string OrderId, string reason)
```

Наименование	Тип	Описание
Cookie	int	Ваш уникальный номер заявки из PlaceOrder
OrderId	string	Id заявки на сервере котировок
Reason	string	Причина

3.9. Событие **OrderSucceeded**

Заявка доставлена на сервер котировок.

Примечание: Для получения этого события необходимо наблюдать за торговым счётом, метод [ListenPortfolio](#).

```
[C#]
```

```
void OrderSucceeded(int cookie, string OrderId)
```

Наименование	Тип	Описание
Cookie	int	Ваш уникальный номер заявки из PlaceOrder
OrderId	string	Id заявки на сервере котировок

3.10. Событие SetPortfolio

Изменился торговый счёт.

Примечание: Для получения этого события необходимо наблюдать за торговым счётом, метод *ListenPortfolio*.

```
[C#]
void SetPortfolio(string portfolio, double cash, double leverage, double comission, double saldo)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Cash	double	Сумма доступных наличных средств на счёте
Leverage	double	Плечё для маржинальной торговли
Commision	double	Сумма биржевой комиссии за торговый день
Saldo	double	Сальдо торгового дня

3.11. Событие UpdateBidAsk

Изменилась очередь заявок по инструменту.

```
[C#]
void UpdateBidAsk(string symbol, int row, int nrows, double bid, double bidsizes, double ask, double asksizes)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Row	int	Порядковый номер строки в очереди заявок
NRows	int	Общее количество строк в очереди заявок
Bid	double	Цена на покупку
BidSize	double	Объем ценных бумаг по цене на покупку
Ask	double	Цена на продажу
AskSize	double	Объем ценных бумаг по цене на продажу

3.12. Событие UpdateOrder

Изменилось состояние приказа.

Примечание: Для получения этого события необходимо наблюдать за торговым счётом, метод *ListenPortfolio*.

```
[C#]
```

```
void UpdateOrder(string portfolio, string symbol, StClientLib.StOrder_State state, StClientLib.StOrder_Action action,
StClientLib.StOrder_Type type, StClientLib.StOrder_Validity validity, double price, double amount, double stop, double
filled, System.DateTime datetime, string orderid, string orderno, , int Status_Mask)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
State	StOrder_State	Состояние приказа. Принимает следующие значения: StOrder_State_ContragentReject=-1 StOrder_State_Submitted=0, Принят ТС StOrder_State_Pending=1, Зарегистрирован в ТС StOrder_State_Open=2, Выведен на рынок StOrder_State_Expired=3, Снят по окончании торгового дня StOrder_State_Cancel=4, Отменён StOrder_State_Filled=5, Исполнен StOrder_State_Partial=6, Частично исполнен StOrder_State_ContragentCancel=7, Отклонён биржей StOrder_State_SystemReject=8, Отклонён биржей StOrder_State_SystemCancel=9, Отклонён биржей
Action	StOrder_Action	Вид торговой операции. Принимает следующие значения: StOrder_Action_Buy = 1, - Купить stOrder_Action_Sell = 2, - Продать stOrder_Action_Short = 3, - Открыть «короткую позицию» StOrder_Action_Cover = 4 – Закрыть «короткую» позицию
Type	StOrder_Type	Тип приказа. Принимает следующие значения: StOrder_Type_Market = 1, - Приказ по рынку StOrder_Type_Limit = 2, - Лимитированный приказ StOrder_Type_Stop = 3, - Стоп приказ StOrder_Type_StopLimit = 4 – приказ Стоп-Лимит
Validity	StOrder_Validity	Срок действия приказа. Принимает следующие значения: StOrder_Validity_Day = 1, - День StOrder_Validity_Gtc = 2 – GTC (до отмены, макс. 30 дней)
Price	double	Цена Лимит, для заявок типа Лимит и Стоп-Лимит)
Amount	Double	Объем, ЦБ в приказе
Stop	Double	Цена СТОП для приказа типа Стоп и Стоп-Лимит
Filled	Double	Объем, оставшийся в приказе
Datetime	DateTime	Время последнего изменения приказа
OrderId	string	Id приказа на сервере котировок
OrderNo	string	Номер приказа на сервере котировок
Status_Mask	int	

3.13. Событие UpdatePosition

Изменилась позиция по инструменту на торговом счёте.

Примечание: Для получения этого события необходимо наблюдать за портфелем, метод [ListenPortfolio](#).

[C#]

```
void UpdatePosition(string portfolio, string symbol, double avprice, double amount, double planned)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
AVPrice	double	Средневзвешенная цена
Amount	double	Объём сделки, если положительный Long, отрицательный в случае Short
Planned	double	Количество ЦБ с учетом выставленных заявок

3.14. Событие UpdateQuote

Изменение текущей котировки на рынке по инструменту.

[C#]

```
void UpdateQuote(string symbol, System.DateTime datetime, double open, double high, double low, double close, double last, double volume, double size, double bid, double ask, double bidsize, double asksize, double open_int, double go_buy, double go_sell, double go_base, double go_base_backed, double high_limit, double low_limit, int trading_status)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Datetime	DateTime	Время последней сделки
Open	double	Цена первой сделки в текущей сессии
High	double	Максимальная цена сделки в текущей сессии
Low	double	Минимальная цена сделки в текущей сессии
Close	double	Цена последней сделки предыдущей торговой сессии
Last	double	Цена последней сделки
Volume	double	Объём по ЦБ за текущую торговую сессию
Size	double	Объём последней сделки
Bid	double	Цена спроса
Ask	double	Цена предложения
BidSize	double	Объём ЦБ с требованием на покупку
AskSize	double	Объём ЦБ с требованием на продажу
Open_Int	double	Совокупная стоимость открытых позиций
Go_Buy	double	Гарантийное обеспечение покупки (фьючерсы)
Go_Sell	double	Гарантийное обеспечение продажи (фьючерсы)
Go_Base	double	Базовое ГО (гарантийное обеспечение продажи опционов)
Go_Base_Backed	double	Гарантийное обеспечение по синтетическим позициям (опционы)
High_Limit	double	Верхний лимит цены
Low_Limit	double	Нижний лимит цены
Trading_Status	int	Недоступно

4. Методы

4.1. Метод `CancelAllOrders`

Отменяет все выставленные приказы на всех площадках.

[C#]

```
void CancelAllOrders()
```

4.2. Метод `CancelBidAsk`

Отменяет получение очереди заявок по инструменту.

[C#]

```
void CancelBidAsks(string symbol)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.3. Метод `CancelOrder`

Отменяет приказ выставленную на рынок методом *PalceOrder*.

[C#]

```
void CancelOrder(string portfolio, string symbol, string orderid)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
OrderId	string	Id приказа на сервере котировок

4.4. Метод `CancelPortfolio`

Отменяет наблюдение за торговым счётом.

[C#]

```
void CancelPortfolio(string portfolio)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре

4.5. Метод CancelQuotes

Отменяет получение котировок по инструменту.

```
[C#]
```

```
void CancelQuotes(string symbol)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.6. Метод CancelTicks

Отменяет получение всех сделок на рынке по инструменту.

```
[C#]
```

```
void CancelTicks(string symbol)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.7. Метод Connect

Вызывает подключение к серверу котировок.

```
[C#]
```

```
void connect(string ip, short port, string login, string password)
```

Наименование	Тип	Описание
IP	string	IP адрес сервера котировок
Port	short	Порт сервера, 8090
Login	string	Логин для доступа к серверу
Password	string	Пароль для доступа

Пример использования:

В примере, демонстрируется использование методов *Connect*, *Disconnect*, *ListenQuotes*, *CancelQuotes*, а также реализован простой способ автореконекта в случае отключения от сервера котировок.

```
[C#]
using System;
using System.Runtime.InteropServices;
using StClientLib;

namespace TestQuotes
{
    class Program
    {
        // SmartCOM
        private static StServer SmartCOM;

        // Для автореконекта
        private static bool bStop;
        private static bool bIsConnect;
        private static bool bConnectIdle;

        private static string sLogin = "ЛОГИН_ДЛЯ_ДОСТУПА_К_СЕРВЕРУ";
        private static string sPassword = "ПАРОЛЬ_К_СЕРВЕРУ";

        private static string[] SelectSymbols = { "GAZR-6.10_FT", "ЛКОН-6.10_FT" };

        static void Main(string[] args)
        {
            bStop = false;
            try
            {
                // Создаём SmartCOM
                SmartCOM = new StServer();
                // Объявляем обработчики для событий
                SmartCOM.Connected += new
                _IStClient_ConnectedEventHandler(SmartCOM_Connected);
                SmartCOM.Disconnected += new
                _IStClient_DisconnectedEventHandler(SmartCOM_Disconnected);
                SmartCOM.UpdateQuote += new
                _IStClient_UpdateQuoteEventHandler(SmartCOM_UpdateQuote);
            }
            catch (Exception Error)
            {
                Console.WriteLine("[SmartCOM]Ошибка, " + Error.Message);
            }
            Connect();
            Console.WriteLine("Press ENTER to quit...");
            Console.ReadLine();
            // Выход
            bStop = true;
            if (SmartCOM != null)
            {
                // Перед выходом рекомендуется отключиться от сервера котировок, если
                // подключены.
                if (bIsConnect)
                {
```

```

        // Передаём команду на отключение
        SmartCOM.disconnect();
        Console.WriteLine("Wait for disconnect.");
        // Ждём отключения.
        while (bIsConnect)
            System.Threading.Thread.Sleep(1000);
    }
    else
        ListenStop();
    Console.WriteLine("Quit...");
    try
    {
        // Освобождаем ресурсы
        SmartCOM.Connected -= SmartCOM_Connected;
        SmartCOM.Disconnected -= SmartCOM_Disconnected;
        SmartCOM.UpdateQuote -= SmartCOM_UpdateQuote;
        if (SmartCOM.GetType().IsCOMObject)
            Marshal.ReleaseComObject(SmartCOM);
        SmartCOM = null;
    }
    catch (Exception Error)
    {
        Console.WriteLine("[SmarCOM] Ошибка при выходе, " + Error.Message);
    }
}

private static void Connect()
{
    bIsConnect = SmartCOM.IsConnected();
    // Проверка на случай если соединение уже установлено.
    if ((SmartCOM != null) && (!bIsConnect))
    {
        bConnectIdle = true;
        Console.WriteLine("[SmartCOM] Coonect to ITServer.");
        // поспим секундочку :-))
        System.Threading.Thread.Sleep(1000);
        if (!bIsConnect)
            SmartCOM.connect("82.204.220.34", 8090, sLogin, sPassword);
    }
    else
        // Если соединение установлено, то начинаем прослушку
        if ((SmartCOM != null) && (bIsConnect))
            ListenStart();
}

private static void ListenStart()
{
    if ((SmartCOM != null) && (bIsConnect))
    {
        Console.WriteLine("[SmartCOM] Подписываемся на данные.");
        foreach (string sSymbol in SelectSymbols)
            SmartCOM.ListenQuotes(sSymbol);
    }
}

private static void ListenStop()
{
    if (SmartCOM != null)
    {

```

```

        Console.WriteLine("[SmartCOM] Отказываемся от получения данных.");
        foreach (string sSymbol in SelectSymbols)
            SmartCOM.CancelQuotes(sSymbol);
    }
}

private static void SmartCOM_Connected()
{
    bConnectIdle = false;
    Console.WriteLine("[SmartCOM] Connected.");
    bIsConnect = SmartCOM.IsConnected();
    // Начинаем прослушку
    ListenStart();
}

private static void SmartCOM_Disconnected(string reason)
{
    Console.WriteLine("[SmartCOM] Disconnected, " + reason);
    // Рекомендуется отказаться от подписанных данных
    ListenStop();
    bIsConnect = SmartCOM.IsConnected();
    if (!bStop && !bConnectIdle)
        Connect();
}

private static void SmartCOM_UpdateQuote(string symbol, System.DateTime datetime,
double open, double high, double low, double close, double last, double volume, double
size, double bid, double ask, double bidsize, double asksize, double open_int, double
go_buy, double go_sell, double go_base, double go_base_backed, double high_limit, double
low_limit, int trading_status)
{
    Console.WriteLine("[Quote:" + symbol + "]" + " " + datetime + " Last: " +
last.ToString() + "[" + size.ToString() + "]" + " GOBuy: " + go_buy.ToString() + "
GOSell:" + go_sell.ToString());
}
}
}

```

4.8. Метод Disconnect

Вызывает отключение от сервера котировок.

[C#]

```
void disconnect()
```

4.9. Метод GetBars

Запрашивает исторические данные по инструменту в интервале.

[C#]

```
int GetBars(string symbol, StClientLib.StBarInterval interval, System.DateTime since, int count)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Interval	StBarInterval	Интервал времени. StBarInterval_1Min = 1, StBarInterval_5Min = 2, StBarInterval_10Min = 3, StBarInterval_15Min = 4, StBarInterval_30Min = 5, StBarInterval_60Min = 6, StBarInterval_2Hour = 7, StBarInterval_4Hour = 8, StBarInterval_Day = 9, StBarInterval_Week = 10, StBarInterval_Month = 11, StBarInterval_Quarter = 12, StBarInterval_Year = 13
Since	DateTime	Дата начала запрашиваемого интервала
Count	int	Количество запрашиваемых интервалов. Если количество запрашиваемых интервалов положительно, сбор идет «назад» по времени в прошлое от указанной даты; если отрицательно – то «вперед»

[C#]

```

using System;
using System.Runtime.InteropServices;
using StClientLib;

namespace AddBarTest
{
    class Program
    {
        private static bool bStop;
        private static bool bConnectIdle;
        static StServer SmartServer;

        static void Main(string[] args)
        {
            try
            {
                SmartServer = new StServer();
                SmartServer.AddBar += new
                _IStClient_AddBarEventHandler(SmartServer_AddBar);
                SmartServer.Connected += new _IStClient_ConnectedEventHandler(Connected);
                SmartServer.Disconnected += new
                _IStClient_DisconnectedEventHandler(Disconnected);
            }
            catch (Exception Error)
            {
                Console.WriteLine("Error: " + Error.Message);
            }
            Connect();
            Console.WriteLine("Press ENTER to Disconnet");
            Console.ReadLine();
        }
    }
}

```

```

        bStop = true;
        SmartServer.disconnect();
        Console.WriteLine("Press ENTER to exit");
        Console.ReadLine();
        if (SmartServer != null)
        {
            Console.WriteLine("Quit...");
            try
            {
                if (SmartServer.GetType().IsCOMObject)
                    Marshal.ReleaseComObject(SmartServer);
            }
            catch (Exception Error)
            {
                Console.WriteLine("Error: " + Error.Message);
            }
        }
    }

    static void SmartServer_AddBar(string symbol, StClientLib.StBarInterval interval,
System.DateTime datetime, double open, double high, double low, double close, double
volume)
    {
        Console.WriteLine("Bar: " + symbol + " " + datetime + " I: " +
interval.GetHashCode() + " O: " + open + " H: " + high + " L: " + low + " C: " + close +
" V: " + volume);
    }

    static void Connected()
    {
        DateTime dtClock = DateTime.Now;
        bConnectIdle = false;
        Console.WriteLine("[Connected] " + dtClock.ToString());
        SmartServer.GetBars("LKOH-6.10_FT", StBarInterval.StBarInterval_10Min,
dtClock, 5);
    }
    static void Disconnected(string Error)
    {
        Console.WriteLine("[Disconnected]");
        if (!bStop && !bConnectIdle)
            Connect();
    }

    private static void Connect()
    {
        // Проверка на случай если соединение уже установлено.
        if (SmartServer != null)
            if (!SmartServer.IsConnected())
            {
                bConnectIdle = true;
                Console.WriteLine("[SmartCOM] Coonect to ITServer.");
                // ПОСПИМ СЕКУНДОЧКУ :- )
                System.Threading.Thread.Sleep(1000);
                if (!SmartServer.IsConnected())
                    SmartServer.connect("82.204.220.34", 8090, "LOGIN", "PASSWORD");
            }
            else
                Connected();
    }
}
}

```

4.10. Метод GetPrortfolioList

Запрашивает получение справочника счетов доступных текущему

[C#]

void GetPortfolioList()

[C#]

```
using System;
using StClientLib;

namespace TestPortfolios
{
    class Program
    {
        private static bool bStop;
        private static bool bConnectIdle;

        static StServer SmartCOM;

        static void Main(string[] args)
        {
            SmartCOM = new StServer();
            SmartCOM.Connected += new _IStClient_ConnectedEventHandler(Connected);
            SmartCOM.Disconnected += new
_IStClient_DisconnectedEventHandler(Disconnected);
            SmartCOM.AddPortfolio += new
_IStClient_AddPortfolioEventHandler(AddPortfolio);
            SmartCOM.SetPortfolio += new
_IStClient_SetPortfolioEventHandler(SetPortfolio);

            Connect();
            Console.ReadLine();
            bStop = true;
            SmartCOM.disconnect();
            Console.WriteLine("Press ENTER to exit");
            Console.ReadLine();
        }

        static void AddPortfolio(int row, int nrows, string portfolioName, string
portfolioExch)
        {
            Console.WriteLine("[AddPortfolio] " + row + "/" + nrows + " " + portfolioName
+ ":" + portfolioExch);
            SmartCOM.ListenPortfolio(portfolioName);
        }

        static void SetPortfolio(string portfolio, double cash, double leverage, double
comission, double saldo)
        {
            Console.WriteLine("[ " + portfolio + " ] Cash: " + cash + " Leverage: " +
leverage + " Fee: " + comission + " Saldo: " + saldo);
        }

        static void Connected()
        {
            bConnectIdle = false;
            Console.WriteLine("[Connected]");
        }
    }
}
```

```

        SmartCOM.GetPrortfolioList();
    }

    static void Disconnected(string Error)
    {
        Console.WriteLine("[Disconnected]");
        if ((!bStop) && (!bConnectIdle))
            Connect();
    }

    private static void Connect()
    {
        // Проверка на случай если соединение уже установлено.
        if (SmartCOM != null)
            if (!SmartCOM.IsConnected())
            {
                bConnectIdle = true;
                Console.WriteLine("[SmartCOM] Coonect to ITServer.");
                // поспим секундочку :- )
                System.Threading.Thread.Sleep(1000);
                if (!SmartCOM.IsConnected())
                    SmartCOM.connect("82.204.220.34", 8090, "LOGIN", "PASSWORD");
            }
            else
                Connected();
    }
}
}

```

4.11. Метод GetSymbols

Запрашивает получение справочника ЦБ

[C#]

void GetSymbols()

[C#]

```

using System;
using System.Runtime.InteropServices;
using System.Globalization;
using System.Collections.Generic;
using StClientLib;

namespace TestSymbols
{
    class Program
    {
        private static List<Tiker> AllTikers;

        private static bool bFirst;
        private static bool bStop;
        private static bool bConnectIdle;
    }
}

```

```

private static StServer SmartCOM;

private static string sLogin = "LOGIN";
private static string sPassword = "PASSWORD";

static void Main(string[] args)
{
    bFirst = true;
    AllTikers = new List<Tiker>();
    bStop = false;
    try
    {
        // Создаём SmartCOM
        SmartCOM = new StServer();
        // Объявляем обработчики для событий
        SmartCOM.Connected += new
_IstClient_ConnectedEventHandler(SmartCOM_Connected);
        SmartCOM.Disconnected += new
_IstClient_DisconnectedEventHandler(SmartCOM_Disconnected);
        SmartCOM.AddSymbol += new
_IstClient_AddSymbolEventHandler(SmartCOM_AddSymbol);
    }
    catch (Exception Error)
    {
        Console.WriteLine("[SmartCOM]Ошибка, " + Error.Message);
    }
    Connect();
    Console.WriteLine("Press ENTER to quit...");
    Console.ReadLine();
    // Выход
    bStop = true;
    if (SmartCOM != null)
    {
        // Перед выходом рекомендуется отключиться от сервера котировок, если
подключены.
        if (SmartCOM.IsConnected())
        {
            // Передаём команду на отключение
            SmartCOM.disconnect();
            Console.WriteLine("Wait for disconnect.");
            // Ждём отключения.
            while (SmartCOM.IsConnected())
                System.Threading.Thread.Sleep(1000);
        }
        Console.WriteLine("Quit...");
        try
        {
            // Освобождаем ресурсы
            SmartCOM.Connected -= SmartCOM_Connected;
            SmartCOM.Disconnected -= SmartCOM_Disconnected;
            SmartCOM.AddSymbol -= SmartCOM_AddSymbol;
            if (SmartCOM.GetType().IsCOMObject)
                Marshal.ReleaseComObject(SmartCOM);
        }
        catch (Exception Error)
        {
            Console.WriteLine("[SmarCOM] Ошибка при выходе, " + Error.Message);
        }
    }
}

```

```

private static void Connect()
{
    // Проверка на случай если соединение уже установлено.
    if ((SmartCOM != null) && (!SmartCOM.IsConnected()))
    {
        bConnectIdle = true;
        Console.WriteLine("[SmartCOM] Coonect to ITServer.");
        // поспим секундочку :-))
        System.Threading.Thread.Sleep(1000);
        if ((SmartCOM != null) && (!SmartCOM.IsConnected()))
            SmartCOM.connect("82.204.220.34", 8090, sLogin, sPassword);
    }
}

private static void SmartCOM_Connected()
{
    bConnectIdle = false;
    Console.WriteLine("[SmartCOM] Connected.");
    // Начинаем прослушку
    if (bFirst)
    {
        bFirst = false;
        // Запрашиваем справочники
        SmartCOM.GetSymbols();
    }
}

private static void SmartCOM_Disconnected(string reason)
{
    Console.WriteLine("[SmartCOM] Disconnected, " + reason);
    if ((!bStop) && (!bConnectIdle))
        Connect();
}

private static void SmartCOM_AddSymbol(int row, int nrows, string symbol, string
short_name, string long_name, string type, int decimals, int lot_size, double punkt,
double step, string sec_ext_id, string sec_exch_name, System.DateTime expiry_date, double
days_before_expiry)
{
    lock (AllTikers)
        AllTikers.Add(new Tiker(symbol, short_name, long_name, step, punkt,
decimals, sec_ext_id, sec_exch_name, expiry_date, days_before_expiry));
    if (row == nrows - 1)
    {
        Console.WriteLine("[SmartCOM] Справочники обновлены");
        TikersOut();
    }
}

private static void TikersOut()
{
    foreach (Tiker tempTiker in AllTikers)
        Console.WriteLine(tempTiker.ToString());
}

public class Tiker
{
    private string sCode;

```

```

private string sShortName;
private string sLongName;
private double dStep;
private double dStepPrice;
private double dDecimals;
private string sSecExtId;
private string sSecExchName;
private System.DateTime dtExpiryDate;
private double dDaysBeforeExpiry;

public Tiker(string code, string shortname, string longname, double step, double
stepprice, double decimals, string sec_ext_id, string sec_exch_name, System.DateTime
expiryDate, double daysbeforeexpiry)
{
    sCode = code;
    sShortName = shortname;
    sLongName = longname;
    dStep = step;
    dStepPrice = stepprice;
    dDecimals = decimals;
    sSecExtId = sec_ext_id;
    sSecExchName = sec_exch_name;
    dtExpiryDate = expiryDate;
    dDaysBeforeExpiry = daysbeforeexpiry;
}

public double ToMoney(double Punkts)
{
    return dStepPrice / dStep * Punkts;
}

public override string ToString()
{
    CultureInfo ci = new CultureInfo("en-us");
    return "[" + sSecExchName + ":" + sCode + "] Step: " + dStep.ToString("G",
ci) + " Money: " + ToMoney(1).ToString("G", ci) + " Short:" + sShortName + " Expiry: " +
dtExpiryDate.ToShortDateString() + "(" + (int)dDaysBeforeExpiry + ")";
}

public string Code { get { return sCode; } }
public string ShortName { get { return sShortName; } }
public string LongName { get { return sLongName; } }
public double Step { get { return dStep; } }
public double StepPrice { get { return dStepPrice; } }
public double Decimals { get { return dDecimals; } }
public string SecExtId { get { return sSecExtId; } }
public string SecExchName { get { return sSecExchName; } }
public System.DateTime ExpiryDate { get { return dtExpiryDate; } }
public double DaysBeforeExpiry { get { return dDaysBeforeExpiry; } }
}
}

```

4.12. Метод IsConnected

Возвращает текущее состояние соединения с сервером котировок.

[C#]

bool **IsConnected**()

Возвращает **true** если соединение установлено, и **false** если связь отсутствует.

Смотри также методы *Connect*, *Disconnect*.

4.13. Метод ListenBidAsks

Позволяет получать очередь заявок по инструменту.

[C#]

void **ListenBidAsks**(**string** *symbol*)

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.14. Метод ListenPortfolio

Позволяет наблюдать за состоянием торгового счёта.

[C#]

void **ListenPortfolio**(**string** *portfolio*)

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре

4.15. Метод ListenQuotes

Позволяет получать котировки по инструменту.

[C#]

void **ListenQuotes**(**string** *symbol*)

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.16. Метод ListenTicks

Позволяет получать все сделки на рынке по инструменту.

[C#]

```
void ListenTicks(string symbol)
```

Наименование	Тип	Описание
Symbol	string	Код ЦБ из таблицы котировок SmartTrade

4.17. Метод MoveOrder

Изменить цену в приказе

[C#]

```
void MoveOrder(string portfolio, string orderno, double targetPrice)
```

Наименование	Тип	Описание
Portfolio	String	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
OrderId	string	Id приказа на сервере котировок
TargetPrce	double	Новая цена для прказа

4.18. Метод PlaceOrder

Выставить приказ на рынок.

[C#]

```
void PlaceOrder(string portfolio, string symbol, StClientLib.StOrder_Action action, StClientLib.StOrder_Type type, StClientLib.StOrder_Validity validity, double price, double amount, double stop, int cookie)
```

Наименование	Тип	Описание
Portfolio	string	Номер торгового счёта на торговой площадке, указывается в верхнем регистре
Symbol	string	Код ЦБ из таблицы котировок SmartTrade
Action	StOrder_Action	Вид торговой операции. Принимает следующие значения: StOrder_Action_Buy = 1, - Купить stOrder_Action_Sell = 2, - Продать stOrder_Action_Short = 3, - Открыть «короткую позицию» StOrder_Action_Cover = 4 – Закреть «короткую» позицию
Type	StOrder_Type	Тип приказа. Принимает следующие значения: StOrder_Type_Market = 1, - Приказ по рынку StOrder_Type_Limit = 2, - Лимитированный приказ StOrder_Type_Stop = 3, - Стоп приказ StOrder_Type_StopLimit = 4 – приказ Стоп-Лимит
Validity	StOrder_Validity	Срок действия приказа. Принимает следующие значения:

		StOrder_Validity_Day = 1, - День StOrder_Validity_Gtc = 2 – GTC (до отмены, макс. 30 дней)
Price	double	Цена Лимит, для заявок типа Лимит и Стоп-Лимит Для приказа: <i>По рынку</i> или <i>Стоп</i> , установите значение 0
Amount	Double	Объем, ЦБ в приказе
Stop	Double	Цена СТОП для приказа типа Стоп и Стоп-Лимит Для приказа: <i>По рынку</i> или <i>Лимит</i> , установите значение 0
Cookie	int	Ваш уникальный номер приказа, используется для определения Id приказа через событие OrderSucceeded, а также для определения ошибки при доставке приказа на сервер котировок, событием OrderFailed.

[C#]

```

using System;
using StClientLib;
using System.Runtime.InteropServices;

namespace TestPlaceOrder
{
    class Program
    {
        static StServer SmartServer;

        static void Main(string[] args)
        {
            try
            {
                SmartServer = new StServer();
                SmartServer.AddTrade += new
                _IStClient_AddTradeEventHandler(SmartServer_AddTrade);
                SmartServer.UpdatePosition += new
                _IStClient_UpdatePositionEventHandler(SmartServer_UpdatePosition);
                SmartServer.UpdateOrder += new
                _IStClient_UpdateOrderEventHandler(SmartServer_UpdateOrder);
                SmartServer.OrderSucceeded += new
                _IStClient_OrderSucceededEventHandler(SmartServer_OrderSucceeded);
                SmartServer.OrderFailed += new
                _IStClient_OrderFailedEventHandler(SmartServer_OrderFailed);
                SmartServer.SetPortfolio += new
                _IStClient_SetPortfolioEventHandler(SmartServer_SetPortfolio);
            }
            catch (Exception Error)
            {
                Console.WriteLine("Error: " + Error.Message);
            }
            Console.WriteLine("IsConnect: " + SmartServer.IsConnected());
            if (!SmartServer.IsConnected())
            {
                SmartServer.connect("82.204.220.34", 8090, "LOGIN", "PASSWORD");
                Console.ReadLine();
            }
            if (SmartServer.IsConnected())
            {
                SmartServer.ListenPortfolio("BPXXXX-RF-01");
                Console.WriteLine("PressKey for PlaceOrder");
            }
        }
    }
}

```

```

        Console.ReadLine();
        SmartServer.PlaceOrder("BPXXXX-RF-01", "GAZR-6.10_FT",
StOrder_Action.StOrder_Action_Sell, StOrder_Type.StOrder_Type_Market,
StOrder_Validity.StOrder_Validity_Day, 0.0d, 1.0d, 0.0d, 12345);
    }
    Console.ReadLine();
    Console.WriteLine("IsConnect: " + SmartServer.IsConnected());
    Console.WriteLine("Press ENTER to quit...");
    Console.ReadLine();
    if (SmartServer != null)
    {
        Console.WriteLine("Quit...");
        try
        {
            SmartServer.CancelPortfolio("BPXXXX-RF-01");
            SmartServer.AddTrade -= SmartServer_AddTrade;
            SmartServer.UpdatePosition -= SmartServer_UpdatePosition;
            SmartServer.UpdateOrder -= SmartServer_UpdateOrder;
            SmartServer.OrderSucceeded -= SmartServer_OrderSucceeded;
            SmartServer.OrderFailed -= SmartServer_OrderFailed;
            SmartServer.SetPortfolio -= SmartServer_SetPortfolio;
            if (SmartServer.GetType().IsCOMObject)
                Marshal.ReleaseComObject(SmartServer);
        }
        catch (Exception Error)
        {
            Console.WriteLine("Error: " + Error.Message);
        }
    }
}

static void SmartServer_SetPortfolio(string portfolio, double cash, double
leverage, double comission, double saldo)
{
    Console.WriteLine("SetPortfolio: " + portfolio + " " + cash + " " + leverage
+ " " + comission + " " + saldo);
}

static void SmartServer_AddTrade(string portfolio, string symbol, string orderid,
double price, double amount, System.DateTime datetime, string tradeno)
{
    Console.WriteLine("AddTrade: " + portfolio + " Symbol: " + symbol + " Id: " +
orderid + " DT: " + datetime + " Price: " + price + " Vol: " + amount + " No: " +
tradeno);
}

static void SmartServer_UpdatePosition(string portfolio, string symbol, double
avprice, double amount, double planned)
{
    Console.WriteLine("UpdatePosition: " + portfolio + " " + symbol + " avprice:
" + avprice + " amount:" + amount + " planned: " + planned);
}

static void SmartServer_UpdateOrder(string portfolio, string symbol,
StClientLib.StOrder_State state, StClientLib.StOrder_Action action,
StClientLib.StOrder_Type type, StClientLib.StOrder_Validity validity, double price,
double amount, double stop, double filled, System.DateTime datetime, string orderid,
string orderno)
{

```

```
        Console.WriteLine("UpdateOrder: " + portfolio + " " + symbol + " " + datetime
+ " " + state + " " + datetime + " Id: " + orderid + " No: " + orderno + " filled: " +
filled + " amount: " + amount);
    }

    static void SmartServer_OrderSucceeded(int cookie, string system_order_id)
    {
        Console.WriteLine("OrderSucceeded: " + cookie + " Id: " + system_order_id);
    }

    static void SmartServer_OrderFailed(int cookie, string system_order_id, string
reason)
    {
        Console.WriteLine("OrderFailed: " + cookie + " Id: " + system_order_id + " "
+ reason);
    }
}
```