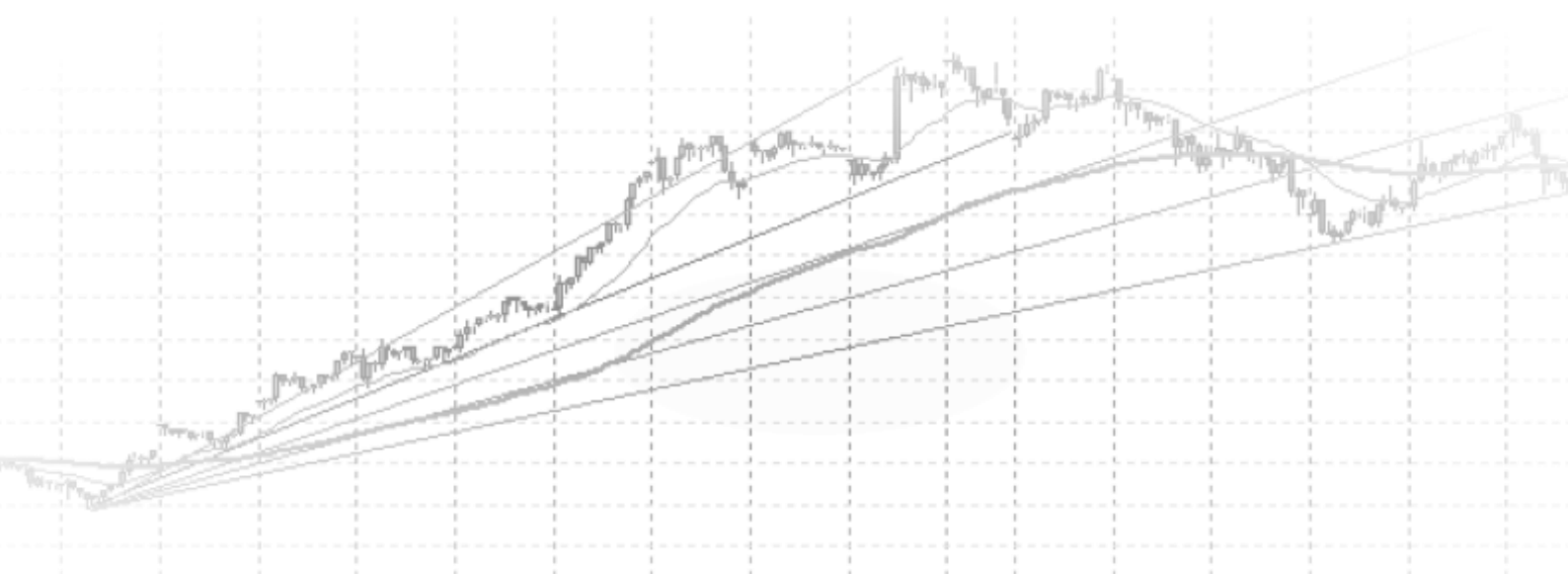




информационные услуги и программные продукты
для инвесторов и профессиональных участников рынка России



Руководство пользователя NIAPI



345.000000 C: 347.420013 V: 2958003

Оглавление

Назначение библиотеки NIAPI	7
Взаимодействие сервера NetInvestor и библиотеки NIAPI	8
Регистрация библиотеки NIAPI	8
Каталог таблиц сервера	9
Структура каталога	9
Типы таблиц	9
Получение данных из таблиц	9
Общее описание классов библиотеки NIAPI	11
Соединение с сервером (SrvSession)	13
Использование криптографии (CryptoStart)	13
Структура данных (StructureClass)	14
Подписка на данные (Table и TableEvents)	14
Контейнеры данных (DataRowSet DataRow)	15
Фильтрация данных (Filter)	15
Отправка транзакций (Transaction)	16
Сообщения (IMessage и IStatusMessage)	16
Примеры использования библиотеки	16
Пример 1. Соединение с сервером	16
Пример 2. Проверка соединения и получение сообщений Status Message	17
Пример 3. Использование криптографии	17
Пример 4. Получение котировок	18
Пример 5. Получение всех сделок	19
Пример 6. Получение всех сделок с фильтром	20
Пример 7. Быстрое получение данных	20
Пример 8. Получение котировок второго порядка «стакана»	20
Пример 9. Отправка заявки	21
Пример 10,11. Получение списка своих заявок и сделок	21
Пример 12, 13. Снятие заявки	22
Пример 14. Получение портфеля и позиций	22
Пример 15, 16. Работа с заявкой стоп-лосс и тейк-профит	23
Просмотр параметров транзакций клиентского места	23
Включение режима журнала в терминале	23
Анализ журнала на значения	23

Приложение. Описание классов библиотеки NIAPI.	25
ISrvrSession.....	25
Свойства.....	25
Методы	25
Connect.....	25
ISrvrSession::Connect	25
ISrvrSession::Disconnect.....	26
ISrvrSession::CryptoStart.....	26
ISrvrSession::CryptoStop	26
ISrvrSession::Ping	26
События	27
_ISrvrSessionEvents::OnLogin	27
_ISrvrSessionEvents:: OnConnectionLost.....	27
_ISrvrSessionEvents:: OnMessage.....	27
_ISrvrSessionEvents:: OnStatusMessage.....	27
IMessage.....	28
Свойства.....	28
IMessage::Command	28
IMessage::Count	28
IMessage::Separator.....	28
IMessage::Value	29
IStatusMessage.....	30
Свойства.....	30
IStatusMessage::Code	30
IStatusMessage::Class	30
IStatusMessage::Description	31
IStatusMessage::ExtendedInfo	31
IStatusMessage::Message	31
Example	32
IStructure.....	33
Свойства.....	33
FieldName.....	33
IStructure::FieldName	33
IStructure::FieldAcronym	34

IStructure::FieldDispName	34
IStructure:: FieldType	34
IStructure2:: FieldCalculate	35
Istructure2:: FieldLinked.....	35
Istructure2:: FieldVisible.....	35
Example	36
IDataRow	37
Свойства.....	37
Example	37
IFilter.....	38
Clear.....	38
Add.....	38
IFilter::Add	38
IDataRowSet	40
Свойства.....	40
GetRow	40
GetRowBSTRArr	40
IDataRowSet::GetField	40
IDataRowSet2::GetRowBSTRArr	40
ITable	42
Свойства.....	42
Методы	43
Open.....	43
GetData.....	43
GetSlice.....	43
SetOnLineUpdates.....	43
AddOnLineFilter	43
DelOnLineFilter	43
Insert.....	43
Modify	43
Delete.....	43
Clear	43
RetrieveField	43
ClearRetrieveFields.....	43

RetrieveUpdateField	43
ClearRetrieveUpdateFields	43
AddUpdateFilter	43
DeleteUpdateFilter	43
OnData	43
_ITableEventsFast_Event_OnData	43
OnDataComplete	43
_ITableEventsFast_Event_OnDataComplete	44
OnInsert	44
_ITableEventsFast_Event_OnInsert	44
OnModify	44
_ITableEventsFast_Event_OnModify	44
OnDelete	44
_ITableEventsFast_Event_OnDelete	44
OnClear	44
_ITableEventsFast_Event_OnClear	44
OnClose	44
_ITableEventsFast_Event_OnClose	44
ITable::Open	44
ITable::GetData	45
ITable::GetSlice	45
ITable::SetOnlineUpdates	45
ITable::Insert	46
ITable::Modify	46
ITable::Delete	46
ITable::Clear	47
ITable::RetrieveField	47
ITable::ClearRetrieveFields	47
ITable::RetrieveUpdateField	47
ITable::ClearRetrieveUpdateFields	47
ITable::AddUpdateFilter	48
ITable::DeleteUpdateFilter	48
_ITableEvents::OnData	49
_ITableEvents::OnDataComplete	49

_ITableEvents::OnClose	49
_ITableEvents::OnInsert	50
_ITableEvents::OnModify	50
_ITableEvents::OnDelete	50
_ITableEvents::OnClear	51
ITransaction	52
Свойства	52
Open	52
Execute	52
OnClose	52
ITransaction::Open	52
ITransaction::Execute	53
_ITransactionEvents::OnClose	53

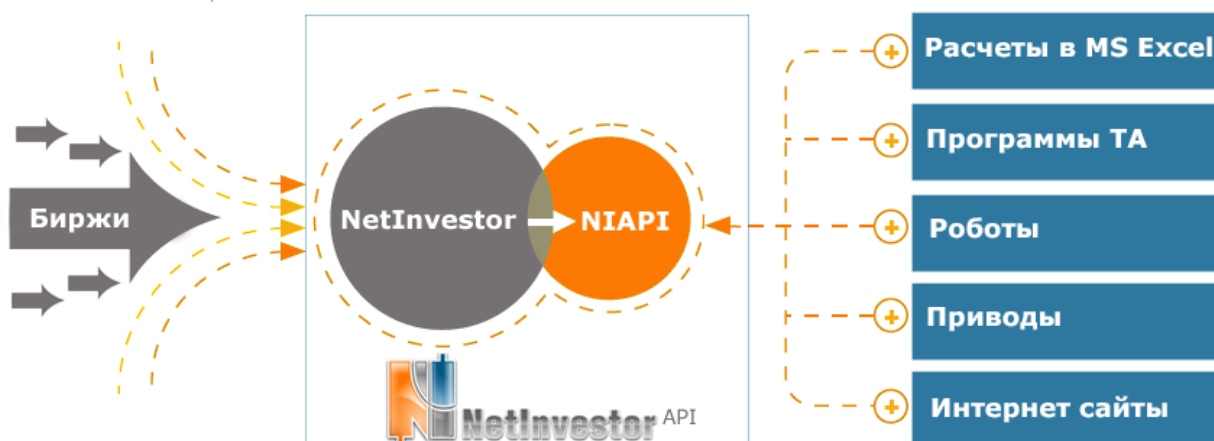
Назначение библиотеки NIAPI

Решение NetInvestor API предлагает программный способ получения торговой информации для расширенной аналитики, а также выставления заявок в торговую систему с высокой скоростью и возможностью обратной связи.

Интерфейс NIAPI позволяет участникам рынка создавать решения, повышающие эффективность работы на рынке. Особенностью NIAPI является наличие всех функций, используемых разработчиками сервера брокера.

Наиболее частые использования библиотеки:

- **Расчеты в Microsoft Excel.** В MS Excel можно поставлять данные прямо с сервера брокера. Такой способ получения данных избавляет от экспортов из торгового терминала и увеличивает скорость поставки данных. Вероятно, главным преимуществом использования MS Excel и NIAPI является возможность выставления заявок, стоп-лоссов, тейк-профитов и осуществления прочих транзакций.
- **Автоматические торговые системы и роботы.** Алгоритмический трейдинг требует скорости в расчете алгоритма и отправления заявки. Используя NIAPI можно написать оптимизированные системы, ограничивая информацию из торговой системы только необходимой и быстрее производя вычисления. Цепочка компонентов в поставке информации в архитектуре робота почти минимальна.
- **Разработка сторонних приложений и приводов.** Реализация «приводов», непосредственно не зависящих от терминала, через NIAPI позволяет получать данные и отправлять заявки непосредственно на сервер.
- **Создание Internet приложений.** Брокерские компании предоставляют Web кабинет с анализом операций клиента и изменениями цен на финансовые инструменты. NIAPI может использоваться для связи с Front-Office системой интернет-трейдинга.
- **Интеграция в системы Back-Office.** Для любых интеграций внутри инфраструктуры брокерских компаний.



Библиотека NetInvestor API осуществляет доставку рыночных данных с различных бирж в сторонние приложения, которым доступны и торговые функции. Информация текущей сессии поставляется в режиме реального времени, а также 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.** Котировочные данные в реальном времени.
- **Все сделки.** Список всех сделок и получение только информации о последней сделке.
- **Котировки второго порядка** (стаканы). Очередь заявок на покупку и продажу.
- **Новости.** Информационная поддержка и новостные ленты.

Торговые транзакции и торговая информация:

- **Счета.** Список счетов клиента.
- **Портфель.** Получение данных по позициям клиента.
- **Заявки.** Выставление всех видов заявок, получение информации по заявкам.
- **Свои сделки.** Получение информации по своим сделкам.
- **Стоп-лоссы.** Выставление стоп-лоссов и получение информации о срабатывании стоп-лоссов.
- **Тейк-профиты.** Выставление тейк-профитов и получение информации о выставлении и результатах срабатывания тейк-профитов.
- **Ценные бумаги.** Список и параметры торгуемых инструментов.

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

Дополнительными преимуществами, которыми пользуются брокерские, инвестиционные и управляющие компаниями:

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

Новости и информационная поддержка. Информационная поддержка МФД-ИнфоЦентр включает новостные ленты ведущих информационных агентств: РИА Новости, Интерфакс, ПРАЙМ-ТАСС, МФД-ИнфоЦентр. Данные с мировых рынков: товары, индексы, валютные пары.

Взаимодействие сервера NetInvestor и библиотеки NIAPI

Библиотека взаимодействует напрямую с сервером брокерской системы, исключая дополнительное звено терминала.



Регистрация библиотеки NIAPI

Прежде чем использовать библиотеку NIAPI, ее необходимо зарегистрировать в системе с помощью утилиты RegSvr32.exe.

Чтобы зарегистрировать NIAPI в MS Windows необходимо выполнить следующие шаги:

- 1) Запустить командную строку из меню «Пуск» - «Программы» - «Стандартные» - «Командная строка» с правами администратора.
- 2) Перейдите в директорию с библиотекой.
- 3) Выполните следующую команду:

```
regsvr32 niapi.dll.
```

Если попытка заканчивается успешно, выводится соответствующее диалоговое окно. В противном случае отображается сообщение об ошибке.

Каталог таблиц сервера

Описание данных сервера: таблиц и транзакций можно посмотреть в мета-описании данных catalog.xml. Каталог содержится на сервере и скачивается при соединении робота либо рабочего места пользователя NetInvestor Professional.

Структура каталога

Каталог содержит следующие разделы:

- Список акронимов. Акронимы представляют собой коды для доступа к информации.
- Список рынков. Список кодов рынков.
- Список таблиц сервера. Список таблиц сервера для подписки на данные.
- Список транзакций. Список транзакций для отправки на биржу.

```
<catalog ver="100520.01" name="ALL">
<acronyms/>
<markets/>
<tables/>
<transactions/>
</catalog>
```

Типы таблиц

Таблицы делятся на два типа:

- адресная (данные таблицы уникальны для пользователя);
- безадресная (данные таблицы являются общими для всех пользователей).

1) Если среди полей таблицы присутствует поле code, то таблица адресная, если нет - то безадресная.

2) В файле catalog.xml находите таблицу или транзакцию(например транзакция ORDER)

```
<ORDER name="Выставление заявки ММББ" type="1" addressed="1">
```

Если есть addressed="1", то таблицу необходимо открывать как адресную.

Получение данных из таблиц

Таблица получает данные на события onData. После окончания получения информации вызывается событие onDataComplete.

Существует три типа обновлений:

- onInsert,
- onModify,
- onDelete.

Для таблиц quotes,quotes_* приходит апдейт только типа on_insert, но работает он не как в других таблицах. В общем случае по данному апдейту возвращается полная строка вставки со всеми полями. А в случае таблиц quotes,quotes_* он работает как неполная вставка, т.е.

приходит событие типа onInsert, но его строка содержит только поля, которые были действительно изменены (это больше похоже на апдейт типа on_modify для других таблиц).

При событии on_modify приходят только ключевые и изменившиеся поля.

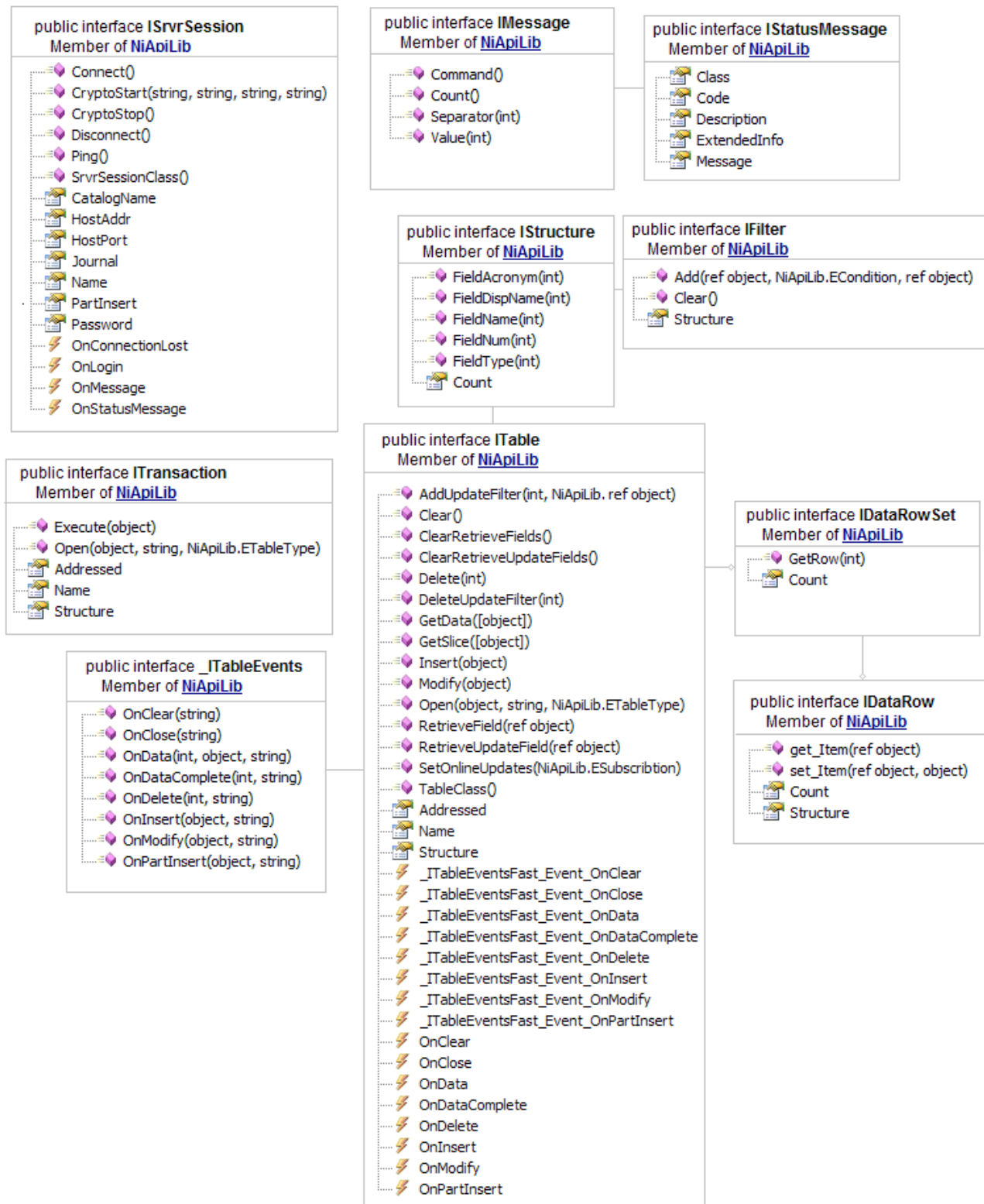
При событии on_delete приходит номер записи (recid), которая была удалена.

Общее описание классов библиотеки NIAPI

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

- **ISrvrSession** – реализация коммуникации и протокольной части, протокольное сжатие, бинарный протокол, передачу и прием защищенных сообщений (ЭЦП и шифрование).
- **IMessage** – интерфейс для работы с протокольными сообщениями. Обмен между сервером и приложением ведется с помощью сообщений и это объект самого низкого уровня.
- **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 транзакций.
- **IDataRowSet** - интерфейс для доступа к коллекциям IDataRow.
- **IFilter** - интерфейс для доступа к коллекциям фильтров для получения исторических данных по таблицам.
- **ITable** – Интерфейс для работы с серверными таблицами, поддерживает описание структуры таблиц, запросы историй, запросы 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 выполнение транзакций.

UML диаграмма классов:



Соединение с сервером (SrvSession)

Для соединения к серверу необходимо знать IP адрес, порт сервера, а также логин с паролем. Соединение с сервером

```
NiApiLib.SrvSession srvrSession = new NiApiLib.SrvSessionClass();
srvrSession.HostAddr = "nidemo.mfd.ru";
srvrSession.HostPort = "2900";
srvrSession.Name = "trader";
srvrSession.Password = "robot";
srvrSession.catalogName = " catalog.xml";
srvrSession.Connect();
...
srvrSession.Disconnect();
```

Запрос на соединение с сервером отправлен, но необходимо обработать ответ от сервера о подключении и других сообщениях. Регистрация обработчиков очень проста:

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

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

Использование криптографии (CryptoStart)

При использовании криптографии необходимо настроить терминал с крипто защитой, прежде чем использовать соединение робота:

Криптография	Файл	Источник
Криптосистема МФД	Nisecure.dll	ООО МФД-ИнфоЦентр
MS Crypto API	NiSec.dll	ООО «Крипто-Про» и ГУП НТЦ «Атлас»
Message Pro	NiSecMP.dll	ЗАО «Сигнал-КОМ»
Верба-OW	NIsecOW.dll	МО ПНИЭИ «Верба»

```
string cryptoLibrary = "NiSec.Dll"; //Имя файла;
string keyBase; //
string userKey;
string serverKey;

srvrSession.CryptoStart(cryptoLibrary, keyBase, userKey, serverKey);
srvrSession.CryptoStop();
```

Следующие параметры могут приниматься в качестве переменных keyBase, userKey, serverKey.

Параметр	Описание
Хранилище	Хранилище сертификата (Крипто-Про, Message-PRO)
ID сертификата пользователя	Сертификат пользователя (Крипто-Про, Message-PRO)
ID сертификата сервера	Сертификат сервера (Крипто-Про, Message-PRO)
Открытые ключи	Открытый ключ (Верба)

Ключевой носитель	Ключевой носитель (Верба)
Номер ключа сервера	Номер ключа (Верба)

Структура данных (StructureClass)

Для получения данных с сервера необходима подписка на таблицы, в которых содержится информация. Описание мета данных таблиц содержится в файле catalog.xml.

Класс StructureClass содержит описание полей таблицы и является вспомогательным при работе с данными таблицы. Его зачастую используют при получении описания данных, либо при подписке на новые данные (Update), либо при подписке на все данные таблицы (GetData).

Описание мета данных таблицы.

```
System.Data.DataTable dtTickers;
NiApiLib.Table tbl;
NiApiLib.Structure myStructure;
NiApiLib.Filter myFiltr;
tbl.Open(this.SrvrSession, "ALL_TRADES", NiApiLib.ETableType.tabAddr);
//Получение структуры
NiApiLib.Structure myStructure=(NiApiLib.Structure)this.tbl.Structure;
for (int i = 0; i < this.myStructure.Count; i++)
{
    //в таблице для отображения данных dtTickers (тип System.Data.DataTable) создаются колонки
    //по количеству полей в строке myrow
    dtTickers.Columns.Add();
    //названия заголовков колонок устанавливаются на значение свойства myStructure.FieldDispName
    //то есть на название полей в таблице данных с сервера
    if (this.myStructure.FieldDispName(i) != "")
    {
        dtTickers.Columns[i].ColumnName = this.myStructure.FieldDispName(i);
    }
}
```

Подписка на данные (Table и TableEvents)

Для подписки на данные необходимо открыть таблицу с данными, указать фильтрацию и способ получения данных

Получение данных из таблицы

```
NiApiLib.Table tbl = new TableClass();
NiApiLib.Structure myStructure = new NiApiLib.StructureClass();
NiApiLib.Filter myFiltr = new FilterClass();
// Создаем обработчики
tbl.OnData += new _ITableEvents_OnDataEventHandler(onData);
tbl.OnDataComplete += new _ITableEvents_OnDataCompleteEventHandler(onDataComplete);
tbl.OnInsert += new _ITableEvents_OnInsertEventHandler(onInsert);
tbl.OnDelete += new _ITableEvents_OnInsertEventHandler(onDelete);
//Открытие таблицы
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);
```

Теперь необходимо реализовать обработчики событий на получение или обновление данных.

```
private void onInsert(object niDataRow, string tableName){}
private void onData(int Code, object niDataRowSet, string tableName){}
private void onDataComplete(int Code, string tableName){}
private void onModify(object niDataRow, string tableName){}
private void onDelete(int Code, string tableName){}
```

Контейнеры данных (DataRowSet DataRow)

Входными параметрами обработчиков событий на получение данных являются контейнеры данных DataRowSet и DataRow.

```
private void onInsert(object niDataRow, string tableName)
{
    if (tableName.Equals("ORDERS ") || tableName.Equals("orders "))
    {
        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();
    }
}
```

Фильтрация данных (Filter)

Технология позволяет ограничивать трафик, используя механизм фильтрации данных в таблицах. Использование фильтра позволяет подписываться только на определенные данные из таблиц системы, указанные в фильтре. Фильтр позволяет создавать логические операции с различными ограничениями на столбцы таблицы.

Установка фильтра на получение данных

```
tbl = new TableClass();
tbl.OnData += new _ITableEvents_OnDataEventHandler(onData);

myStructure = new NiApiLib.StructureClass();
myFiltr = new FilterClass();

tbl.Open(session.getNISrvrSession(), "ALL_TRADES ", NiApiLib.ETableType.tabNoneAddr);
myFiltr.Structure = this.tbl.Structure;

object o1 = (object)"I_NAME";
object o2 = (object)ticker;
myFiltr.Add(ref o1, NiApiLib.ECondition.conEqual, ref o2);

this.tbl.SetOnlineUpdates(NiApiLib.ESubscription.subEnabled);
```

Установка фильтра на получение изменений

Для установки получения изменений данных необходимо при получении данных определить условия и фильтры для таблицы.

Завершение передачи данных из таблицы осуществляется в методе onDataComplete. Далее должен быть определен фильтр для таблицы по одному или нескольким желаемым полям.

```
private void onDataComplete(int Code, string tableName)
{
    object o1 = "I_NAME";
    object o2 = ticker;
    tbl.AddUpdateFilter(0, ELogicalOperation.logAnd, ref o1, NiApiLib.ECondition.conEqual, ref o2);
    tbl.SetOnlineUpdates(ESubscription.subEnabled);
}
```

Отправка транзакций (Transaction)

Транзакции необходимы для отправления приказов на биржу, например, отправление заявки. Поскольку тейк-профиты, стоп-лоссы хранятся на сервере брокерской системы, то транзакциями они не являются.

```
NiApiLib.SrvrSession Session = new NiApiLib.SrvrSessionClass();
NiApiLib.Transaction Trans = new NiApiLib.TransactionClass();
NiApiLib.DataRow row = new NiApiLib.DataRowClass();

...
Trans.Open(Session, "ORDER", ETableType.tabAdrr); // Открываем транзакцию
row.Structure = Trans.Structure; //интерфейс для работы с описанием форматов транзакций.
object ACCOUNT = "I_ACCOUNT";
row.set_Item(ref ACCOUNT, "S01-00000F00"); // Устанавливаем параметры
...
Trans.Execute(row); // Отправляем запрос на выполнение транзакции
```

Сообщения (IMessage и IStatusMessage)

В сообщениях дополнительно сообщаются служебные сообщения от сервера.

```
SMessage = (NiApiLib.StatusMessage)pStatusMessage;
mes = (NiApiLib.Message)SMessage.Message;
sr.Write(SMessage.ExtendedInfo + " | " + SMessage.Class );
sr.WriteLine(" ");
for (int i = 0; i < mes.Count()-1; i++)
{ sr.WriteLine("    " + mes.Value(i) + " | " + mes.Separator(i));}
```

Примеры использования библиотеки

К документации присоединены примеры, которые можно скомпилировать и использовать в своих работах. Ниже мы в примерах описаны основные ключевые моменты при работе с наиболее часто встречающимися примерами.

Пример 1. Соединение с сервером

В данном примере рассматривается соединение с сервером NetInvestor .

В данном коде мы рассматриваем пример запуска класса, который осуществляет соединение и разрыв соединения с сервером.

```
using System;
using System.Collections.Generic;
using System.Linq;
using NiApiLib;

namespace Connection
{
    class Program
    {
        static void Main(string[] args)
        {
            Connection c = new Connection();
            c.connect();
            Console.ReadLine();
            c.disconnect();
        }
    }
}
```

1. Необходимо объявить объект NiApiLib.SrvrSessionClass, отвечающий за соединение с сервером, присвоить ему параметры соединения. Полный список параметров находится в описании объекта Свойства интерфейса ISrvrSession.

```
class Connection
{
    NiApiLib.SrvrSession Session = new NiApiLib.SrvrSessionClass();
```

```
public void connect()
{
    Session.HostAddr = "nidemo.mfd.ru";
    Session.HostPort = 2900;
    Session.Name = "test";
    Session.Password = "12345678";
    Session.CatalogName = "catalog.xml";
    Session.OnLogin+=new _ISrvrSessionEvents_OnLoginEventHandler(Session_OnLogin);
    Session.Connect();
}
```

II. Необходимо реализовать метод Disconnect

```
public void disconnect()
{
    Session.Disconnect();
}
```

III. Последним этапом является обработчик события OnLogin(int), отвечающего за обработку соединения с сервером.

```
public void Session_OnLogin(int LoginResult) {}
}
```

В реальном работе необходимо реализовать и другие обработчики, как например, обработчик разрыва связи `_ISrvrSessionEvents:: OnConnectionLost`. Для подробного получения перечня свойств и методов обратитесь к описанию объекта `ISrvrSession`.

Пример 2. Проверка соединения и получение сообщений Status Message

Для проверки соединения с сервером используется метод `ISrvrSession.Ping()`. В примере ответ сервера принимается в объект `NiApiLib.StatusMessage` и выводится на консоль.

I. Объект `NiApiLib.SrvrSessionClass`, отвечающий за соединение с сервером, объявляется как в предыдущем примере. Аналогично реализуются методы `Connect()` и `Disconnect()`.

II. Объявляется обработчик события `_ISrvrSessionEvents:: OnConnectionLost`, который при успешном соединении (`LoginResult = 0`) использует метод `ISrvrSession.Ping()` для получения ответа сервера.

```
public void Session_OnLogin(int LoginResult) {
    if (LoginResult == 0) {
        Session.Ping();
    } else {
        Console.WriteLine("[System] ошибка подключения");
    }
}
```

III. Объявляется обработчик события `_ISrvrSessionEvents:: OnConnectionLost`. При получении нового статусного сообщения, ответ читается с помощью свойства `NiApiLib.StatusMessage.Message`. Текст сообщения выводится на консоль.

```
void Session_OnStatusMessage(object pStatusMessage){}
```

Внимание! Для корректной работы торговой платформы и для снижения трафика пользовательской программы, рекомендуем пользоваться методом `Ping` только в случае отсутствия обновления данных. Поступающие обновления от сервера гарантируют корректное соединение с сервером.

Пример 3. Использование криптографии

При использовании криптографии необходимо отслеживать исключения для получения информации о состоянии криптозащиты.

I. Для реализации криптографии необходимо выполнить метод `ISrvrSession::CryptoStart` передавая ему соответствующие параметры. Необходимо также обработать исключения возможные при работе с криптографией.

```
try
{
    this.srvrSession.CryptoStart("cryptoLibrary", "keyBase", "userKey", "serverKey");
}
catch (System.ArgumentException eCryptInvArg)
{
    Console.WriteLine("Неверные параметры криптографии.\n" + eCryptInvArg.Message);
}
catch (System.FieldAccessException eFaildArg)
{
    Console.WriteLine("Ошибка при активации криптографии." + eFaildArg.Message);
}
catch (System.NotImplementedException eNotEmpl)
{
    Console.WriteLine("Криптография не была инициализирована" + eNotEmpl.Message);
}
catch (System.Exception exp)
{
    Console.WriteLine("Ошибка при запуске криптографии" + exp.Message);
}
```

II. При завершении работы необходимо завершить работу криптографии методом `ISrvrSession::CryptoStop`.

```
try
{
    this.srvrSession.CryptoStop();
}
catch (System.UnauthorizedAccessException eCrypt)
{
    Console.WriteLine("Crypto hasn't been initilized" + eCrypt.Message);
}
```

Пример 4. Получение котировок

Необходимо объявить таблицу котировок: (ММББ КЦБ: QUOTES, FORTS Фьючерсы: QUOTES_PSFU, FORTS Опционы: QUOTES_PSOP)

I. Необходимо объявить следующие объекты для работы с таблицей котировок: `ITable`, `IFilter`, `IDataRow`.

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();
NiApiLib.Filter Flt = new NiApiLib.FilterClass();
NiApiLib.DataRowSet DrowSet= new NiApiLib.DataRowSetClass();
NiApiLib.DataRow row = new NiApiLib.DataRowClass();
```

II. Необходимо объявить обработчики на получение данных и обновления данных

```
Tbl.OnInsert+=new _ITableEvents_OnInsertEventHandler(Tbl_OnInsert);
Tbl.OnData+=new _ITableEvents_OnDataEventHandler(Tbl_OnData);
```

III. Необходимо открыть таблицу для получения данных, установить признак получения данных

```
Tbl.Open(Session, "QUOTES", ETableType.tabNoneAddr);
Tbl.SetOnlineUpdates(ESubscription.subEnabled);
Flt.Structure = Tbl.Structure;
Tbl.GetSlice(Flt);
```

IV. Реализуем обработчик получения данных котировок

```
void Tbl_OnData(int Code,object DataRowSet, string TableName)
{
    DrowSet = (NiApiLib.DataRowSet)DataRowSet;
    for (int get_i=0; get_i <= DrowSet.Count-1; get_i++)
    {
        row = (NiApiLib.DataRow)DrowSet.GetRow(get_i);
        for (int get_j = 0; get_j <= row.Count-1; get_j++)
```

```

        {
            object get_ondate = get_j;
            Console.WriteLine(row.get_Item(ref get_ondate) + " | ");
        }
        Console.WriteLine(" \n\n");
    }
}

```

V. Реализуем обработчик обновления котировок

```

void Tbl_OnInsert(object NiDataRow, string TableName)
{
    row = (NiApiLib.DataRow)NiDataRow;
    Console.WriteLine(" [Update] ");
    for (int ins_i=0; ins_i<=row.Count-1;ins_i++)
    {
        object get_insert = ins_i;
        Console.WriteLine(row.get_Item(ref get_insert) + " | ");
    }
    Console.WriteLine(" ");
}

```

Пример 5. Получение всех сделок

Данные всех сделок можно получать из таблиц для ММББ КЦБ: ALL_TRADES, для FORTS Фьючерсы: ALL_TRADES_PSFU, FORTS Опционы: ALL_TRADES_PSOP.

I. Для получения всех сделок необходимо как на получение любой таблицы объявить классы: ITable,

IFilter, IDataRowSet, IDDataRow.

```

NiApiLib.SrvrSession Session = new NiApiLib.SrvrSessionClass();
NiApiLib.Table Tbl = new NiApiLib.TableClass();
NiApiLib.Filter Flt = new NiApiLib.FilterClass();
NiApiLib.DataRowSet DrowSet= new NiApiLib.DataRowSetClass();
NiApiLib.DataRow row = new NiApiLib.DataRowClass();

```

II. Объявляем один из обработчиков для реализации

```

Tbl.OnData+=new _ITableEvents_OnDataEventHandler(Tbl_OnData);

```

III. Открываем таблицу для получения данных

```

Tbl.Open(Session, "ALL_TRADES", ETableType.tabNoneAddr);
Flt.Structure = Tbl.Structure;
Tbl.GetData(Flt);

```

IV. Пишем обработчик получения данных, накопленных за торговую сессию

```

public void Tbl_OnData(int Code,object DataRowSet, string TableName)
{
    DrowSet = (NiApiLib.DataRowSet)DataRowSet;
    for (int get_i=0; get_i <= DrowSet.Count-1; get_i++)
    {
        Console.WriteLine(" [OnData] ");
        row = (NiApiLib.DataRow)DrowSet.GetRow(get_i);
        for (int get_j = 0; get_j <= row.Count-1; get_j++)
        {
            object get_ondate = get_j;
            Console.WriteLine(row.get_Item(ref get_ondate) + " | ");
        }
        Console.WriteLine(" \n\n");
    }
}

```

V. Пишем обработчик получения обновлений таблицы Всех сделок

```

public void Tbl_OnInsert(object DataRow, string TableName)
{
    row = (NiApiLib.DataRow)DataRow;
    Console.WriteLine(" [OnInsert] ");
    for (int ins_i=0; ins_i<=row.Count-1;ins_i++)
    {
        object get_insert = ins_i;

```

```
Console.WriteLine(row.get_Item(ref get_insert) + " | ");  
}  
Console.WriteLine(" ");  
}
```

Внимание! Для корректной работы торговой платформы и для снижения трафика пользовательской программы, рекомендуем получать таблицы методом OnData и накапливать данные методом OnInsert. При этом слияние и хранение данных на протяжении торговой сессии должен обеспечивать процесс пользователя.

Пример 6. Получение всех сделок с фильтром

Использование фильтра для получения данных помогает получать данные только по указанным критериям, например, только по указанным инструментам. Фильтрация осуществляется с помощью интерфейса

IFilter.

I. Объявление переменных и установка обновлений в таблицу

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();  
NiApiLib.Filter Flt = new NiApiLib.FilterClass();  
Tbl.SetOnlineUpdates(ESubscription.subEnabled);
```

II. Определение фильтра для желаемого акронима.

```
object NAME = "I_NAME";  
object RESULT = "ЛУКОЙЛ";
```

III. Установка фильтра для получения обновлений onInsert

```
Tbl.AddUpdateFilter(0, 0, ref NAME, ECondition.conEqual, ref RESULT);
```

IV. Установка фильтра на получение данных onData.

```
Flt.Structure = Tbl.Structure;  
Flt.Add(ref NAME, ECondition.conEqual, ref RESULT);  
Tbl.GetSlice(Flt);
```

Пример 7. Быстрое получение данных

Для более быстрого получения данных (через SafeArray) используются события _ITableEventsFast. Установка соединения, открытие таблиц и работа с фильтрами аналогичны предыдущим примерам.

I. Объявление обработчиков событий для «быстрого» получения данных.

```
public void connect()  
{  
    tbl._ITableEventsFast_Event_OnData+=new _ITableEventsFast_OnDataEventHandler(Fast_Event_OnData);  
    tbl._ITableEventsFast_Event_OnInsert+=new _ITableEventsFast_OnInsertEventHandler(tbl _ITableEventsFast_Event_OnInsert);  
}
```

II. Реализация обработчика события _ITableEventsFast_Event_OnData (получение всех данных).

```
public void Fast_Event_OnData(int Code, object DataRowSet, string TableName){}
```

III. Реализация обработчика события _ITableEventsFast_Event_OnInsert (получение обновлений).

```
public void tbl _ITableEventsFast_Event_OnInsert(object datarow, string tablename){}
```

Пример 8. Получение котировок второго порядка «стакана»

Для получения котировок второго порядка необходима таблица ORDERBOOK для рынка ММББ КЦБ. Для рынка FORTS Фьючерсы и Опционы, соответственно ORDERBOOK_PSFU и ORDERBOOK_PSFU.

I. Необходимо объявить переменные таблиц, фильтров и контейнеров данных.

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();  
NiApiLib.Filter Flt = new NiApiLib.FilterClass();  
NiApiLib.DataRowSet DrowSet= new NiApiLib.DataRowSetClass();  
NiApiLib.DataRow row = new NiApiLib.DataRowClass();
```

II. Объявляем обработчики для таблицы

```
Tbl.OnInsert+=new ITableEvents_OnInsertEventHandler(Tbl_OnInsert);
Tbl.OnModify+=new ITableEvents_OnModifyEventHandler(Tbl_OnModify);
Tbl.OnDelete+=new ITableEvents_OnDeleteEventHandler(Tbl_OnDelete);
Tbl.OnData+=new ITableEvents_OnDataEventHandler(Tbl_OnData);
```

III. Выполняем обработку всех четырех обработчиков:

```
public void Tbl_OnData(int Code,object DataRowSet, string TableName)
{
    Console.WriteLine(" [OnData] ");
}
public void Tbl_OnInsert(object NiDataRow, string TableName)
{
    Console.WriteLine(" [OnInsert] ");
}
public void Tbl_OnModify(object NiDataRow, string TableName)
{
    Console.WriteLine(" [OnModify] ");
}
public void Tbl_OnDelete(int Code, string TableName)
{
    Console.WriteLine("[OnDelete] "+Code);
}
```

Пример 9. Отправка заявки

Отправление заявки осуществляется посредством отправки транзакции на биржу. Выбор транзакции осуществляется в каталоге. Транзакцию имплементирует следующий объект: ITransaction.

Рассмотрим пример выставления заявки на рынке FORTS Фьючерсы:

I. Объявление переменной транзакции и контейнера данных DataRow.

```
NiApiLib.Transaction trans = new NiApiLib.TransactionClass();
NiApiLib.DataRow row = new NiApiLib.DataRowClass();
```

II. Открытие транзакции с указанием названия транзакции

```
trans.Open(Session, "FUTADDORDER", ETableType.tabAdrr);
row.Structure = trans.Structure;
```

III. Установка параметров для транзакции

```
object SECCODE="I_SECCODE";
row.set_Item(ref SECCODE,"RTS-6.10"); // Код инструмента
object BUYSSELL="I_BUYSSELL";
row.set_Item(ref BUYSSELL,"B"); // Направление: B – покупка, S – продажа
object QUANTITY="I_QUANTITY";
row.set_Item(ref QUANTITY,"2"); //Количество
object PRICE="I_PRICE";
row.set_Item(ref PRICE, "132255"); // Цена
object MKTLIMIT="I_MKTLIMIT";
row.set_Item(ref MKTLIMIT,"M"); //Тип заявки: M - котировочная, L - встречная
object BROKERREF="I_BROKERREF";
row.set_Item(ref BROKERREF,"552"); //Идентификатор клиента
object SECBOARD="I_SECBOARD";
row.set_Item(ref SECBOARD, "PSFU"); // Рынок: PSFU - FORTS Фьючерсы, PSOP - FORTS Опционы
object ACCOUNT="I_ACCOUNT";
row.set_Item(ref ACCOUNT, "AC1"); // Счет
```

IV. Отправка транзакции

```
trans.Execute(row);
```

Пример 10,11. Получение списка своих заявок и сделок

Получение заявок осуществляется из таблиц ORDERS*, например, ORDERS для ММББ КЦБ, ORDERS_PSFU для FORTS Фьючерсы и ORDERS_PSOP для FORTS Опционов.

Получение своих сделок осуществляется из таблиц DEALS*, DEALS для ММББ КЦБ, DEALS_PSFU для FORTS Фьючерсы и ORDERS_PSOP для FORTS Опционов и прочих. Полный перечень рынков и таблиц находится в каталоге.

I. Начинаем с объявления переменных типа ITable, IFilter, IDataRow.

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();  
NiApiLib.Filter Flt = new NiApiLib.FilterClass();  
NiApiLib.DataRowSet DrowSet= new NiApiLib.DataRowSetClass();  
NiApiLib.DataRow row = new NiApiLib.DataRowClass();
```

II. Устанавливаем обработчики на существующие и новые сделки

```
Tbl.OnInsert+=new _ITableEvents_OnInsertEventHandler(Tbl_OnInsert);  
Tbl.OnData+=new _ITableEvents_OnDataEventHandler(Tbl_OnData);
```

III. Открываем таблицу и включаем необходимые обновления

```
Tbl.Open(Session, "DEALS", ETableType.tabAddr);  
Tbl.SetOnlineUpdates(ESubscription.subEnabled);  
Flt.Structure = Tbl.Structure;  
Tbl.GetData(Flt);
```

IV. Добавляем обработчики событий на новые и существующие записи

```
Tbl.OnInsert+=new _ITableEvents_OnInsertEventHandler(Tbl_OnInsert);  
Tbl.OnData+=new _ITableEvents_OnDataEventHandler(Tbl_OnData);
```

V. Выполняем обработку данных непосредственно в обработчиках

```
public void Tbl_OnData(int Code,object DataRowSet, string TableName){ }  
public void Tbl_OnInsert(object NiDataRow, string TableName) { }
```

Пример 12, 13. Снятие заявки

Для снятия заявки на рынке FORTS Фьючерсы выбираем в каталоге транзакцию для снятия заявки:

```
Trans.Open(Session, "FUTDEORDER", ETableType.tabAddr);  
Console.WriteLine("[System] Транзакция открыта");  
Row.Structure = Trans.Structure;  
object ORDER_NUMBER="I_ORDER_NUMBER";  
Row.set_Item(ref ORDER_NUMBER, "183346580"); // № записи I_RID  
object ACCOUNT="I_ACCOUNT";  
Row.set_Item(ref ACCOUNT, "AC1"); // Счет  
Trans.Execute(Row);  
Console.WriteLine("[System] Снятие заявки");
```

Для снятия заявки на бирже ММББ:

```
Trans.Open(Session, "WD_ORDER_BY_NUMBER", ETableType.tabAddr);  
Row.Structure = Trans.Structure;  
object ORDER_NUMBER="I_ORDER_NUMBER";  
Row.set_Item(ref ORDER_NUMBER, 1111);  
Trans.Execute(Row);
```

Пример 14. Получение портфеля и позиций

Портфель можно получить из таблиц PORTFOLIO*. Для рынка ММББ КЦБ PORTFOLIO, для рынка FORTS Фьючерсы и Опционы с PORTFOLIO_PSFU и PORTFOLIO_PSOP.

I. Начинаем с объявления таблицы, фильтра и контейнеров данных: ITable, IFilter, IDataRow.

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();  
NiApiLib.Filter Flt = new NiApiLib.FilterClass();  
NiApiLib.DataRowSet DrowSet= new NiApiLib.DataRowSetClass();  
NiApiLib.DataRow row = new NiApiLib.DataRowClass();
```

II. Добавляем обработчики событий делегаты:

```
Tbl.OnInsert+=new _ITableEvents_OnInsertEventHandler(Tbl_OnInsert);  
Tbl.OnModify+=new _ITableEvents_OnModifyEventHandler(Tbl_OnModify);  
Tbl.OnData+=new _ITableEvents_OnDataEventHandler(Tbl_OnData);
```

III. Открываем таблицу на получение данных и обновления

```
Tbl.Open(Session, "PORTFOLIO", ETableType.tabAddr);
Console.WriteLine("[System] Таблица открыта");
Tbl.SetOnlineUpdates(ESubscription.subEnabled);
Flt.Structure = Tbl.Structure;
Tbl.GetData(Flt);
```

IV. Пишем обработчики

```
public void Tbl_OnData(int Code, object DataRowSet, string TableName) {}
public void Tbl_OnInsert(object NiDataRow, string TableName) {}
public void Tbl_OnModify(object NiDataRow, string TableName) {}
```

Пример 15, 16. Работа с заявкой стоп-лосс и тейк-профит

Выставление стоп-лосса осуществляется на таблицах типа STOPLOSS*.

I. Объявляем переменные и открываем таблицу

```
NiApiLib.Table Tbl = new NiApiLib.TableClass();
NiApiLib.DataRow Row = new NiApiLib.DataRowClass();
Tbl.Open(Session, "STOPLOSS", ETableType.tabAddr);
Tbl.SetOnlineUpdates(ESubscription.subEnabled);
Row.Structure = Tbl.Structure;
```

II. Устанавливаем параметры

```
object SHORTNAME="I SHORTNAME";
Row.set_Item(ref SHORTNAME, "+МосЭнерго"); // Инструмент
object DIRECTION="I_DIRECTION";
Row.set_Item(ref DIRECTION, "3"); // Выставление на: 3-покупка, 4-продажа
object PRICE="I_PRICE";
Row.set_Item(ref PRICE, "3.30"); // Цена
object PRICESIZE="I_PRICESIZE";
Row.set_Item(ref PRICESIZE, "1"); // Количество
object NAME="I_NAME";
Row.set_Item(ref NAME, "MSNG"); // Тикер
object MARKET_LEVEL="I_MARKET_LEVEL";
Row.set_Item(ref MARKET_LEVEL, "3.30"); // Уровень рынка для стоп-лосса
```

III. Выставляем стоп-лосс командой **Insert**

```
Tbl.Insert(Row);
```

Просмотр параметров транзакций клиентского места

Для просмотра параметров отправляемых торговым терминалом NetInvestor Professional можно воспользоваться дополнительным режимом с включенными журналом. В этом режиме терминал записывает все входящие и исходящие сообщения в текстовый файл. Открыв файл журнала можно посмотреть какие параметры отправляет терминал при определенных операциях.

Включение режима журнала в терминале

Для включения режима журнала необходимо выключить терминал. Перейти в директорию установки NIPro и открыть файл: «\Config\nipro_config\config.xml». Отредактируйте XML файл установив значение атрибута \NiAdmin\Server\Connection\@Journal = 1 включите терминал, который в директории установки будет автоматически создавать файлы журнала при каждом подключении.

Анализ журнала на значения

Начало файла выглядит следующим образом:

```
15.06.2010 17:12:13 Version - 2.0.5.931
15.06.2010 17:12:13 Connecting...
15.06.2010 17:12:13 OK
15.06.2010 17:12:13 In:  L 200↔2004 L
15.06.2010 17:12:13 Out: *login/password*
15.06.2010 17:12:13 In:  L 200↔1 L
```

```
15.06.2010 17:12:13 Out:  L 100↔1000L
15.06.2010 17:12:13 In:  L 200↔101▼15062010L
15.06.2010 17:12:13 In:  L 200↔200▼1L
15.06.2010 17:12:13 In:  L 200↔201▼nitestL
15.06.2010 17:12:13 In:  L 200↔400L
15.06.2010 17:12:13 In:  L 200↔2005▼nullL
```

Завершение файла при разрыве связи:

```
16.06.2010 15:54:19 SHUTDOWN: Initiated
16.06.2010 15:54:19 ShutDown:
16.06.2010 15:54:19 Tcp connection interrupted by EventShutdown
```

В файл пишутся два типа сообщений:

- Входящие сообщения. Помечаются после времени кодом In:
- Исходящие сообщения. Помечаются после времени кодом Out:

Пример входящего сообщения:

```
16.06.2010 15:54:18 In:
L 5↔QUOTES▼1482499▲0▼1482499▲1▼ЛУКОЙЛ▲2▼15:54:21▲38▼EQBR▲39▼LКОН▲5▼4545▲9▼4850▲27▼16.0
6.2010L
```

Пример исходящего сообщения:

```
17.06.2010 11:38:28 Out:
L 500↔ni▼ci.ci.ni▼1↔1↔2↔STOPLOSS▼3▼3▲5▼1▲10▼LКОН▲11▼1707.40▲12▼/ci▲13▼33▲14▼48▲15▼1▲17
▼L01-00000F00▲20▼119▲22▼EQBR▲23▼EQBR▲25▼LКОН▲26▼135▲27▼138L
```

В сообщениях указывается, время, имя таблицы, следуя за логином пользователя и перечисление отправляемых или получаемых полей.

Нотация полей и их значений выглядит следующим образом: ▲НОМЕР ПОЛЯ▼ЗНАЧЕНИЕ .

Пример:

▲10▼LКОН

Открыв описание таблиц каталога catalog.xml таблицу STOPLOSS можно найти номера всех значений:

```
<STOPLOSS name="Список стоп-лоссов/алертов ММВБ КЦБ" type="14" addressed="1" lowercase="1" name_en="Stop loss orders
MICEX Corp.">
  <struct>
    <I_RID num="0" visible="1"/>
    <I_CLIENT_NAME num="1"/>
    <I_SHORTNAME num="2"/>
    <I_DIRECTION num="3"/>
    <I_PRICE num="4"/>
    <I_PRICESIZE num="5"/>
    <I_ALERT_LEVEL num="6"/>
    <I_ORDER_CONDITION num="7"/>
    <I_TIME num="8"/>
    <I_DATE num="9"/>
    <I_NAME num="10"/>
    ...
  </struct>
</STOPLOSS>
```

Исходя из данных каталога: описания таблицы и акронимов получаем однозначное соответствие номеру поля и его значению.

```
<I_NAME num="10"/>
▲10▼LКОН
<I_NAME fieldtype="2" length="30" name="Тикер" visible="1" name_en="Ticker name" cache="1"/>
```

Приложение. Описание классов библиотеки NIAPI.

ISrvrSession

Реализация коммуникации и протокольной части, поддерживает работу через прокси службы, протокольное сжатие, бинарный протокол, передачу и прием защищенных сообщений (ЭЦП и шифрование).

Внимание! В качестве имени сервера зашито “netinvestor”

Свойства

BSTR HostAddr	Адрес сервера	
LONG HostPort	Порт сервера	
BSTR Name	Логин клиента	
BSTR Password	Пароль клиента	
BSTR CatalogName	Путь к файлу с каталогом (xml)	
ProxyConfig		(не реализовано)
UserCertificate		(не реализовано)
HostCertificate		(не реализовано)
BSTR Journal	Путь к файлу журнала (лога)	

Методы

Connect	Установление соединения
Disconnect	Разрыв соединения
CryptoStart	Включение системы криптозащиты
CryptoStop	Выключение системы криптозащиты
Ping	Тестирование соединения с сервером
События	
OnLogin	Вызывается при аутентификации клиента, в ответ на вызов метода Connect
OnConnectionLost	Вызывается при потере соединения с сервером
OnMessage	Вызывается при приходе сообщения от сервера
OnStatusMessage	Вызывается при приходе статусного сообщения от сервера

ISrvrSession::Connect

HRESULT Connect(void);

Return values

S_OK	Успешное выполнение
------	---------------------

ISrvrSession::Disconnect

```
HRESULT Disconnect(void);
```

Return values

S_OK	Успешное выполнение
------	---------------------

ISrvrSession::CryptoStart

```
HRESULT CryptoStart(  
    [in] BSTR FileName,  
    [in] BSTR BaseName,  
    [in] BSTR NodeName,  
    [in] BSTR RemoteName  
);
```

Parameters

FileName	Путь к файлу библиотеки криптозащиты
BaseName	База
NodeName	Идентификатор пользователя
RemoteName	Идентификатор сервера

Return values

S_OK	Успешное выполнение
E_ACCESSDENIED	Библиотека криптозащиты уже инициализирована
E_INVALIDARG	Ошибка загрузки библиотеки криптозащиты
E_NOTIMPL	В библиотеке криптозащиты не найдены необходимые функции
E_FAIL	Ошибка создания криптоконтекста

ISrvrSession::CryptoStop

```
HRESULT CryptoStop(void);
```

Return values

S_OK	Успешное выполнение
E_ACCESSDENIED	Библиотека криптозащиты не инициализирована
E_NOTIMPL	В библиотеке криптозащиты не найдены необходимые функции
E_FAIL	Ошибка выгрузки библиотеки криптозащиты

ISrvrSession::Ping

```
HRESULT Ping(void);
```

Return values

S_OK	Успешное выполнение
------	---------------------

События

_ISrvrSessionEvents::OnLogin

```
HRESULT OnLogin(  
    [in] LONG LoginResult  
);
```

Parameters:

LoginResult 0 - Авторизация успешна
 1 - Ошибка авторизации

_ISrvrSessionEvents:: OnConnectionLost

```
HRESULT OnConnectionLost(void);
```

_ISrvrSessionEvents:: OnMessage

```
HRESULT OnMessage(  
    [in] IDispatch* Msg  
);
```

Parameters:

Msg Указатель на сообщение

_ISrvrSessionEvents:: OnStatusMessage

```
HRESULT OnStatusMessage(  
    [in] IDispatch* pStatusMessage  
);
```

Parameters:

pStatusMessage Указатель на сообщение

IMessage

Интерфейс для работы с протокольными сообщениями

Свойства

LONG Count	Число полей в сообщении. Свойство доступно только для чтения.
LONG Command	Команда по протоколу. Свойство доступно только для чтения.
LONG* Separator	Символ-разделитель. Свойство доступно только для чтения.
VARIANT* Value	Значение поля. Свойство доступно только для чтения.

IMessage::Command

```
HRESULT Command(  
    [out,retval] LONG* Value  
);
```

Parameters

Value	Команда
-------	---------

Return values

S_OK	Успешное выполнение
------	---------------------

IMessage::Count

```
HRESULT Count(  
    [out,retval] LONG* Value  
);
```

Parameters

Value	Число полей в сообщении
-------	-------------------------

Return values

S_OK	Успешное выполнение
------	---------------------

IMessage::Separator

```
HRESULT Separator(  
    [in] LONG Index,  
    [out,retval] LONG* Value  
);
```

Parameters

Index	Номер поля в сообщении
Value	Значение разделителя для поля

Return values

S_OK Успешное выполнение

IMessage::Value

```
HRESULT Value(  
    [in] LONG Index,  
    [out,retval] VARIANT* Val  
);
```

Parameters

Index	Номер поля в сообщении
Value	Значение поля

Return values

S_OK Успешное выполнение

IStatusMessage

Интерфейс для работы с протокольными статусными сообщениями и сообщениями об ошибках.

Свойства

LONG Code	Код сообщения. Свойство доступно только для чтения.
BSTR Class	Класс сообщения. Свойство доступно только для чтения.
BSTR Description	Текст сообщения (второе поле сообщения, если полей больше трех). Используется для некоторых типов серверных сообщений. Свойство доступно только для чтения.
BSTR ExtendedInfo	Дополнительная информация (Разделенные ';' поля сообщения от третьего и далее). Свойство доступно только для чтения.
Idispatch* Message	Интерфейс к данным сообщения в формате IMessage. Необходимо использовать в том случае, когда число полей в сообщении больше единицы. Свойство доступно только для чтения.

IStatusMessage::Code

```
HRESULT Code(  
    [out, retval] LONG* pVal  
);
```

Parameters

pVal Код сообщения

Return values

S_OK Успешное выполнение
E_NOTIMPL Сообщение не проинициализировано.

IStatusMessage::Class

```
HRESULT Class(  
    [out, retval] BSTR* pVal  
);
```

Parameters

pVal Класс сообщения

Return values

S_OK Успешное выполнение
E_NOTIMPL Сообщение не проинициализировано.

IStatusMessage::Description

```
HRESULT Description(  
    [out, retval] BSTR* pVal  
);
```

Parameters

pVal Текст сообщения (первое поле сообщения).

Return values

S_OK Успешное выполнение
E_NOTIMPL Сообщение не проинициализировано.

IStatusMessage::ExtendedInfo

```
HRESULT ExtendedInfo (  
    [out, retval] BSTR* pVal  
);
```

Parameters

pVal Дополнительная информация.

Return values

S_OK Успешное выполнение
E_NOTIMPL Сообщение не проинициализировано.

IStatusMessage::Message

```
HRESULT Message (  
    [out, retval] IDispatch** pVal  
);
```

Parameters

pVal Интерфейс к данным сообщения в формате IMessage. Необходимо использовать в том случае, когда число полей в сообщении больше единицы.

Return values

S_OK Успешное выполнение
E_NOTIMPL Сообщение не проинициализировано.

Example

```
//процедура обработки статусных сообщений
Sub session_OnStatusMessage( Msg )
    //выводим информацию о статусном сообщении
    WScript.Echo " StatusMessage( [" & Msg.Code &"], " & Msg.Description &
        ", " & Msg.Class & ", " & Msg.ExtendedInfo & ")"

    //получаем свойство Message и выводим информацию о сообщении
    For i=0 to (Msg.Message.Count-1)
        WScript.Echo "    [" & i & "] : " & Msg.Message.Separator(i) & " :
            " & Msg.Message.Value(i)
    Next

    //при приходе сообщения 2006 разрываем соединение с сервером
    If (Msg.Code = 2006) Then
        SrvrSession.Disconnect
    End If
End Sub
```

IStructure

Интерфейс для работы с описанием структур таблиц и форматов транзакций.

Добавлен интерфейс IStructure2. Класс NiApiLib.StructureClass теперь содержит интерфейсы IStructure и IStructure2.

Свойства

LONG Count	Число полей в структуре. Свойство доступно только для чтения.
LONG Version	Версия. (не реализовано)
LONG Type	Тип структуры (не реализовано)

Методы

IStructure

Получение имени поля

FieldName

FieldAcronym	Получение акронима поля
FieldType	Получение типа поля
FieldDispName	Получение имени поля для отображения

Методы

IStructure2

FieldCalculate	Сообщает, вычисляемое поле или нет (0/1)
FieldLinked	Сообщает, ссылочное поле (1/0)
FieldVisible	Сообщает, видимое поле или нет в интерфейсе пользователя.(1/0)

IStructure::FieldName

```
HRESULT FieldName(  
    [in] LONG Index,  
    [out, retval] BSTR* pVal  
);
```

Parameters

Index	Номер поля в структуре
PVal	В случае успешного выполнения функции - указатель на строку с именем поля

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Неверный номер поля в структуре

IStructure::FieldAcronym

```
HRESULT FieldAcronym(  
    [in] LONG Index,  
    [out, retval] BSTR* pVal  
);
```

Parameters

Index	Номер поля в структуре
PVal	В случае успешного выполнения функции - указатель на строку с акронимом поля

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Неверный номер поля в структуре

IStructure::FieldDispName

```
HRESULT FieldDispName(  
    [in] LONG Index,  
    [out, retval] BSTR* pVal  
);
```

Parameters

Index	Номер поля в структуре
PVal	В случае успешного выполнения функции - указатель на строку с именем поля для отображения

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Неверный номер поля в структуре

IStructure::FieldType

```
HRESULT FieldType(  
    [in] LONG Index,  
    [out, retval] LONG* pVal  
);
```

Parameters

Index	Номер поля в структуре
PVal	В случае успешного выполнения функции – заполняется типом поля

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Неверный номер поля в структуре

IStructure2:: FieldCalculate

```
HRESULT FieldCalculate(  
    [in] LONG i,  
    [out,retval] LONG* pNum  
);
```

Parameters

i Номер поля в структуре
pNum 0 – невычислимое, иначе – код вычисления

Return values

S_OK Успешное выполнение
E_INVALIDARG Неверный номер поля в структуре

IStructure2:: FieldLinked

```
HRESULT FieldLinked(  
    [in] LONG i,  
    [out,retval] LONG* pNum  
);
```

Parameters

i Номер поля в структуре
pNum 0 – не ссылочное поле, иначе – код словаря

Return values

S_OK Успешное выполнение
E_INVALIDARG Неверный номер поля в структуре

IStructure2:: FieldVisible

```
HRESULT FieldVisible(  
    [in] LONG i,  
    [out,retval] LONG* pNum  
);
```

Parameters

i Номер поля в структуре
pNum 0 – невидимое, 1 – видимое поле

Return values

S_OK Успешное выполнение
E_INVALIDARG Неверный номер поля в структуре

Example

```
Dim Counter
Dim SrvrSession
Dim Quotes
Dim Structure

//создаем объект сессии
Set SrvrSession = WScript.CreateObject( "NiApi.SrvrSession" , "session_" )

//заполняем параметры соединения
SrvrSession.HostAddr = "194.87.225.162"
SrvrSession.HostPort = 2900
SrvrSession.Name     = "username"
SrvrSession.Password = "password"
//устанавливаем соединение с сервером
SrvrSession.Connect
Connected = 1
Do While Connected <> 0
    WScript.Sleep 250
Loop

//разрыв соединения
SrvrSession.Disconnect

//освобождаем объекты таблицы и сессии
Quotes      = Null
SrvrSession = Null

Sub session_OnLogin( Result )
    //создаем объект таблицы
    Set Quotes = WScript.CreateObject( "NiApi.Table" , "quotes_" )
    //открываем таблицу котировок
    Quotes.Open SrvrSession, "QUOTES"
    //отображаем информацию о каждом поле структуры
    For i=0 To (Quotes.Structure.Count-1)
        WScript.Echo Quotes.Structure.FieldDispName(i)
        WScript.Echo Quotes.Structure.FieldName(i)
        WScript.Echo Quotes.Structure.FieldAcronym(i)
        WScript.Echo Quotes.Structure.FieldType(i)
    Next
End Sub
```

IDataRow

Интерфейс для доступа к «полям записи» таблиц и данным транзакций.

Свойства

LONG	Count	Число полей в строке. Свойство доступно только для чтения
VARIANT	Item	Элемент строки.
IDispatch	Structure	Интерфейс структуры таблицы Istructure.

Example

```
'процедура обработки потока входных данных по таблице
Sub table_OnData( Code, DataRowSet )
    For i=0 To (DataRowSet.Count-1)
        'вывод поля по имени
        WScript.Echo DataRowSet.GetRow(i).Item("RECID")
        'вывод поля по акрониму
        WScript.Echo DataRowSet.GetRow(i).Item("I_NAME")
        'вывод поля по номеру
        WScript.Echo DataRowSet.GetRow(i).Item(5)
    Next
End Sub
```

IFilter

Интерфейс для доступа к коллекциям фильтров для получения исторических данных по таблицам.

Свойства

IDispatch Structure Интерфейс структуры таблицы Istructure.

Методы

Clear

Очистка всех имеющихся фильтров в коллекции

Add

Добавление фильтра в коллекцию

IFilter::Add

```
HRESULT Add(
    [in] VARIANT* pIndexVar,
    [in] ECondition Condition,
    [in] VARIANT* pVal
);
```

Parameters

pIndexVar Идентификатор поля, по которому ставится фильтр. (VT_I4 или VT_BSTR). Если тип VT_I4, то переменная должна содержать номер поля в структуре таблицы. Если тип VT_BSTR, то переменная должна содержать имя поля или акроним.

Condition Условие для фильтра. Принимает следующие значения

```
conEqual            - равно,
conNotEqual        - не равно,
conLess            - меньше,
conLessOrEqual     - меньше либо равно,
conGreater         - больше,
conGreaterOrEqual - больше либо равно.
```

pVal Значение поле фильтра

Return values

S_OK Успешное выполнение

E_INVALIDARG Значение номера поля выходит за допустимые пределы (в случае добавления фильтра по номеру поля), либо неизвестное имя поля или акроним (в случае добавления фильтра по ним), либо неподдерживаемый тип для переменной pIndexVar.

E_NOTIMPL Возникает при добавления фильтра по имени инструмента или акрониму в том случае, если не установлено значение свойства Structure.

Example

```
Set Quotes = WScript.CreateObject( "NiApi.Table" , "quotes_" )
Quotes.Open SrvrSession, "QUOTES", 0
WScript.Echo "Quotes fields count " & Quotes.Structure.Count
Set Filter = WScript.CreateObject( "NiApi.Filter" )
Filter.Structure = Quotes.Structure
Filter.Add "I_NAME",0,"PAO EЭC"
num = Quotes.GetSlice(Filter)
```

IDataRowSet

Интерфейс для доступа к коллекциям **IDataRow**. В

Добавлен интерфейс IDataRowSet2 для скоростного доступа к записям в случае работы с большими объемами данных

Свойства

LONG Count Число строк. Свойство доступно только для чтения.

Методы IDataRowSet

GetRow

Получение строки

Методы IDataRowSet2

GetRowBSTRArr

Получение строки в виде SafeArray из BSTR

IDataRowSet::GetField

```
HRESULT GetRow(  
    [in] LONG Index,  
    [out, retval] IDispatch** pRow  
);
```

Parameters

Index Номер поля в структуре
pRow В случае успешного выполнения функции – указатель на интерфейс

Return values

S_OK Успешное выполнение
E_INVALIDARG Значение Index выходит за допустимые пределы

IDataRowSet2::GetRowBSTRArr

```
HRESULT GetRowBSTRArr (  
    [in] LONG Index,  
    [out, retval] Variant* pRow  
);
```

Parameters

Index Номер поля в структуре
pRow В случае успешного выполнения функции – массив полей SafeArray

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Значение Index выходит за допустимые пределы
E_OUTOFMEMORY	Ошибка выделения памяти для массива
E_FAIL	Данные не записаны

ITable

Интерфейс для работы с серверными таблицами, поддерживает описание структуры таблиц, запросы историй, запросы срезов, запросы на вставку, модификацию и удаление записей, управление подпиской на обновления. События реентерабельны.

Свойства

BSTR Name

Число строк. Свойство доступно только для чтения.

ETableType Addressed

Определяет тип таблицы. Для адресной таблицы принимает значение “tabAddr”, для безадресной – “tabNoneAddr”

Idispatch Structure

Интерфейс структуры, связанной с таблицей. Свойство доступно только для чтения

Методы

Open

GetData

GetSlice

SetOnLineUpdates

AddOnLineFilter

DelOnLineFilter

Insert

Modify

Delete

Clear

RetrieveField

ClearRetrieveFields

RetrieveUpdateField

ClearRetrieveUpdateFields

AddUpdateFilter

DeleteUpdateFilter

Открытие таблицы

Отправка запроса данных по таблице

Отправка запроса среза по таблице

Разрешение или запрещение синхронизации данных в таблице и на сервере в реальном времени

(не реализовано)

(не реализовано)

Отправка запроса на добавление строки в таблицу

Отправка запроса на изменение строки в таблице

Отправка запроса на удаление строки в таблице

Отправка запроса на очистку таблицы

Добавление поля в список полей, приходящих с сервера в истории

Очистка списка полей, приходящих с сервера в истории

Добавление поля в список полей, приходящих с сервера в апдейтах

Очистка списка полей, приходящих с сервера в апдейтах

Добавление фильтра на изменение данных по таблице

Удаление фильтра на изменение данных по таблице

События

OnData

_ITableEventsFast_Event_OnData

OnDataComplete

Вызывается при получении блока данных в ответ на вызов **GetData** или **GetSlice**

Событие аналогично OnData с разницей в том, что данные передаются через SafeArray, в результате чего отсутствует необходимость преобразования типов из IDispatch.

Вызывается по окончании получения всех данных, приходящих с сервера в ответ на вызов **GetData** или **GetSlice**

`_ITableEventsFast_Event_OnDataComplete`

Событие аналогично `OnDataComplete` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

`OnInsert`

Вызывается при получении от сервера уведомления об операции добавления данных в таблицу

`_ITableEventsFast_Event_OnInsert`

Событие аналогично `OnInsert` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

`OnModify`

Вызывается при получении от сервера уведомления об операции модификации данных в таблицы

`_ITableEventsFast_Event_OnModify`

Событие аналогично `OnModify` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

`OnDelete`

Вызывается при получении от сервера уведомления об операции удаления данных из таблицы

`_ITableEventsFast_Event_OnDelete`

Событие аналогично `OnDataDelete` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

`OnClear`

Вызывается при получении от сервера уведомления о завершении операции очистки таблицы в ответ на запрос **Clear**

`_ITableEventsFast_Event_OnClear`

Событие аналогично `OnClear` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

`OnClose`

Вызывается при потере соединения с сервером. Служит для индикации того, что `ISrvrSession` уничтожил ссылку на объект таблицы.

`_ITableEventsFast_Event_OnClose`

Событие аналогично `OnClose` с разницей в том, что данные передаются через `SafeArray`, в результате чего отсутствует необходимость преобразования типов из `IDispatch`.

ITable::Open

```
HRESULT Open(  
    [in] IDispatch* SrvrSession,  
    [in] BSTR Name,  
    [in] ETableType Type  
);
```

Parameters

SrvrSession	Указатель на интерфейс открытой сессии
Name	Имя таблицы
Type	Тип таблицы (ADDRESSED или NONADDRESSED)

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::GetData

```
HRESULT GetData(  
    [in, optional] IDispatch* Filter,  
    [out, retval] LONG Code  
);
```

Parameters

Filter	Интерфейс на список фильтров
Code	Уникальный код запроса

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::GetSlice

```
HRESULT GetSlice(  
    [in, optional] IDispatch* Filter,  
    [out, retval] LONG Code  
);
```

Parameters

Filter	Интерфейс на список фильтров
Code	Уникальный код запроса

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::SetOnlineUpdates

```
HRESULT SetOnlineUpdates(  
    [in] ESubscription Val  
);
```

Parameters

Val	SubEnabled для разрешения подписки на обновления и
-----	--

subDisabled для запрещения

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::Insert

```
HRESULT Insert(  
    [in] IDispatch* NiDataRow  
);
```

Parameters

NiDataRow	Указатель на интерфейс IDataRow с данными
-----------	---

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::Modify

```
HRESULT Modify(  
    [in] IDispatch* NiDataRow  
);
```

Parameters

NiDataRow	Указатель на интерфейс IDataRow с данными
-----------	---

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::Delete

```
HRESULT Delete(  
    [in] LONG Recid  
);
```

Parameters

Recid	Уникальный идентификатор удаляемой записи
-------	---

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::Clear

```
HRESULT Clear();
```

Return values

S_OK	Успешное выполнение
E_FAIL	Таблица не открыта

ITable::RetrieveField

```
HRESULT RetrieveField(  
    [in] VARIANT* Field  
);
```

Parameters

Field	Номер поля по протоколу, акроним или имя поля в базе данных на сервере
-------	--

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Некорректное значение поля Field

ITable::ClearRetrieveFields

```
HRESULT ClearRetrieveFields();
```

Return values

S_OK	Успешное выполнение
------	---------------------

ITable::RetrieveUpdateField

```
HRESULT RetrieveUpdateField(  
    [in] VARIANT* Field  
);
```

Parameters

Field	Номер поля по протоколу, акроним или имя поля в базе данных на сервере
-------	--

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Некорректное значение поля Field

ITable::ClearRetrieveUpdateFields

```
HRESULT ClearRetrieveUpdateFields();
```

Return values

S_OK Успешное выполнение

ITable::AddUpdateFilter

```
HRESULT AddUpdateFilter(
    [in] LONG ParentId,
    [in] ELogicalOperation Operation,
    [in] VARIANT* Field,
    [in] ECondition Condition,
    [in] VARIANT* Value,
    [out, retval] LONG* FilterId
);
```

Parameters

ParentId	Идентификатор родительского фильтра. Корневой идентификатор равен 0.
Operation	Логическое условие для добавления к родительскому фильтру (logAnd, logOr)
Field	Номер поля по протоколу, акроним или имя поля в базе данных на сервере
Condition	Условие для поля фильтрации (conEqual, conNotEqual, conLess, conLessOrEqual, conGreater, conGreaterOrEqual)
Value	Значение поля для фильтрации
FilterId	В случае успешного завершения функции содержит идентификатор добавленного фильтра

Return values

S_OK	Успешное выполнение
E_INVALIDARG	Некорректное значение поля Field, Condition, Value или Operation
E_FAIL	Ошибка отправки запроса на сервер

ITable::DeleteUpdateFilter

```
HRESULT OnData(
    [in] LONG FilterId,
);
```

Parameters

FilterId	Идентификатор удаляемого фильтра. Автоматически удаляются все дочерние фильтры. Для удаления всех фильтров по таблице достаточно удалить корневой (FilterId=0)
----------	--

Return values

S_OK Успешное выполнение

E_FAIL

Ошибка отправки запроса на сервер

События

_ITableEvents::OnData

```
HRESULT OnData(  
    [in] LONG Code,  
    [in] IDispatch* DataRowSet,  
    [in] BSTR TableName  
);
```

Parameters

Code	Уникальный код запроса
DataRowSet	Указатель на интерфейс IRowSet с данными
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

_ITableEvents::OnDataComplete

```
HRESULT OnDataComplete(  
    [in] LONG Code,  
    [in] BSTR TableName  
);
```

Parameters

Code	Уникальный код запроса
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

_ITableEvents::OnClose

```
HRESULT OnClose(  
    [in] BSTR TableName  
);
```

Parameters

TableName	Имя таблицы
-----------	-------------

Return values

S_OK

Успешное выполнение

_ITableEvents::OnInsert

```
HRESULT OnInsert(  
    [in] IDispatch* NiDataRow,  
    [in] BSTR TableName  
);
```

Parameters

NiDataRow	Указатель на интерфейс IDataRow с данными
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

_ITableEvents::OnModify

```
HRESULT OnModify(  
    [in] IDispatch* NiDataRow,  
    [in] BSTR TableName  
);
```

Parameters

NiDataRow	Указатель на интерфейс IDataRow с данными
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

_ITableEvents::OnDelete

```
HRESULT OnDelete(  
    [in] LONG Recid,  
    [in] BSTR TableName  
);
```

Parameters

Recid	Уникальный идентификатор удаляемой записи
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

_ITableEvents::OnClear

```
HRESULT OnClear(  
    [in] BSTR TableName  
);
```

Parameters

Recid	Уникальный идентификатор удаляемой записи
TableName	Имя таблицы

Return values

S_OK	Успешное выполнение
------	---------------------

ITransaction

Интерфейс для работы с серверными транзакциями, поддерживает описание структуры транзакции и запросы на выполнение транзакций.

Свойства

BSTR Name	Число строк. Свойство доступно только для чтения.
ETableType Addressed	Определяет тип транзакции. Для адресной принимает значение “tabAddr”, для безадресной – “tabNoneAddr”
IDispatch Structure	Интерфейс структуры, связанной с транзакцией. Свойство доступно только для чтения

Методы

Open

Открытие транзакции

Execute

Отправка запроса на выполнение транзакции

События

OnClose

Вызывается при потере соединения с сервером. Служит для индикации того, что ISrvrSession уничтожил ссылку на объект таблицы.

ITransaction::Open

```
HRESULT Open(  
    [in] IDispatch* SrvrSession,  
    [in] BSTR Name,  
    [in] ETableType Type  
);
```

Parameters

SrvrSession	Указатель на интерфейс открытой сессии
Name	Имя таблицы
Type	Тип таблицы (ADDRESSED или NONADDRESSED)

Return values

S_OK	Успешное выполнение
E_FAIL	Транзакция не открыта

ITransaction::Execute

```
HRESULT Execute(  
    [in] IDispatch* NiDataRow  
);
```

Parameters

NiDataRow Указатель на интерфейс IDataRow с данными

Return values

S_OK Успешное выполнение

E_FAIL Таблица не открыта

События

_ITransactionEvents::OnClose

```
HRESULT OnClose();
```

Return values

S_OK Успешное выполнение

Example

```
Set ChangePortfolio = WScript.CreateObject( "NiApi.Transaction" , "transaction_" )  
ChangePortfolio.Open SrvrSession,"NI_CHANGEPORTFOLIO", 0  
Set Row = WScript.CreateObject( "NiApi.DataRow" )  
Row.Structure = ChangePortfolio.Structure  
Row.Item(0) = 1120  
Row.Item(1) = "EQBR"  
Row.Item(2) = "EESR"  
Row.Item(6) = 1  
Row.Item(3) = 2    'тип инструмента  
Row.Item(4) = 44 'направление изменения  
Row.Item(5) = 100 'размер изменения  
ChangePortfolio.Execute (Row)  
Row                = Null  
ChangePortfolio   = Null
```