ПРОГРАММНЫЙ ИНТЕРФЕЙС ПОДКЛЮЧЕНИЯ ВНЕШНИХ СИСТЕМ К ТОРГОВО-КЛИРИНГОВОЙ СИСТЕМЕ ММВБ

(Библиотека MTESRL.DLL, v. 4.0)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 2

БИБЛИОТЕКА MTESRL.DLL 2

СЦЕНАРИЙ РАБОТЫ С БИБЛИОТЕКОЙ 2

РЕГИСТРАЦИЯ НА СЕРВЕРЕ 3

ПОЛУЧЕНИЕ СЛУЖЕБНОЙ И СЕРВИСНОЙ ИНФОРМАЦИИ 5

- ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ШЛЮЗЕ 5
- ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ВЕРСИИ КЛИЕНТСКОЙ БИБЛИОТЕКИ 6

ПОЛУЧЕНИЕ ОПИСАНИЯ ИНФОРМАЦИОННЫХ ОБЪЕКТОВ 7

РАБОТА С ИНФОРМАЦИОННЫМИ ОБЪЕКТАМИ 7

- ВЫПОЛНЕНИЕ ТРАНЗАКЦИЙ 8
- РАБОТА С ТАБЛИЦАМИ 9
  - Открытие таблицы 9
  - Запрос изменений 10
  - Закрытие таблицы 11
  - Пример работы с таблицами 11
  - Замечания по работе с таблицами 12

ОПТИМИЗАЦИЯ ИСПОЛЬЗОВАНИЯ ПАМЯТИ 13

ВОССТАНОВЛЕНИЕ ПОСЛЕ СБОЯ НА MICEX BRIDGE SERVER 13

- ПРИМЕР ВОССТАНОВЛЕНИЯ ПОСЛЕ СБОЯ 15

ЗАВЕРШЕНИЕ СЕАНСА СВЯЗИ 15

СООБЩЕНИЯ ОБ ОШИБКАХ 15

КОДЫ ОШИБОК 16

ПРИЛОЖЕНИЕ 1. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTESTRUCTURE 17

ПРИЛОЖЕНИЕ 2. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTEOPENTABLE 20

ПРИЛОЖЕНИЕ 3. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTEREFRESH 21

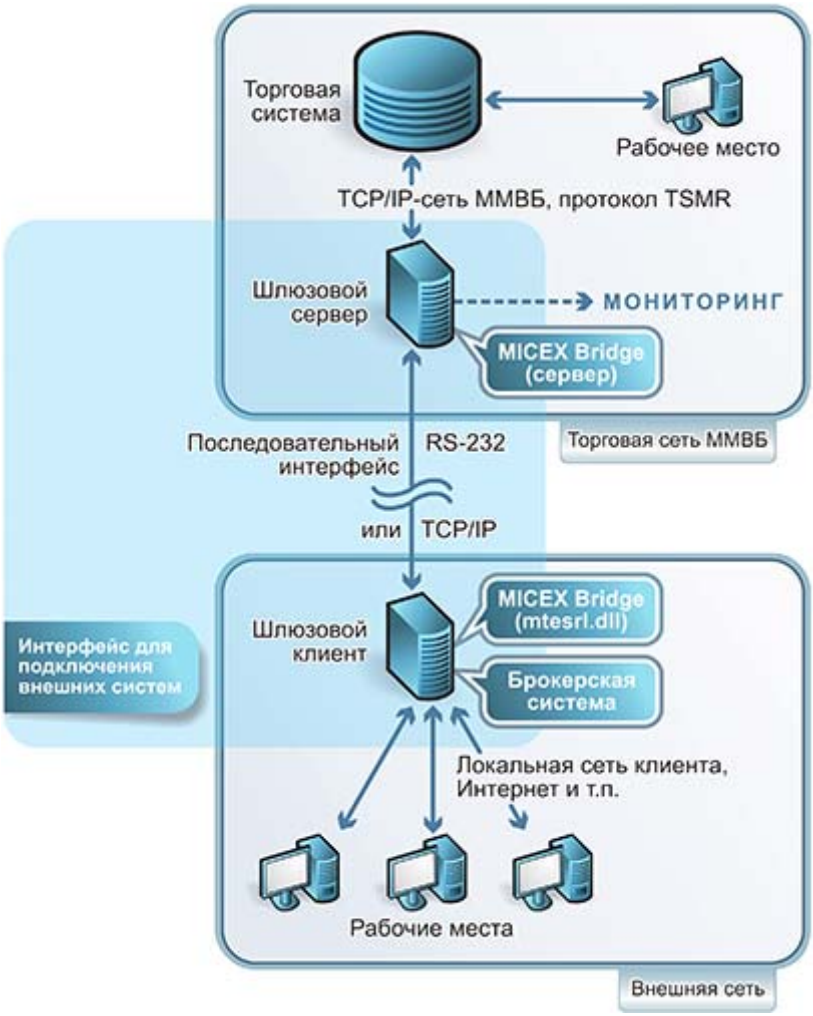
ПРИЛОЖЕНИЕ 4. ЭЛЕМЕНТАРНЫЕ ТИПЫ 21

ПРИЛОЖЕНИЕ 5. ФОРМАТИРОВАНИЕ ПОЛЕЙ С ТИПАМИ FLOAT, FIXED, DATE, TIME 22

Введение

Программный интерфейс позволяет подключать к торгово-клиринговой системе ММВБ внешние системы распространения торговой информации, сбора клиентских заявок, ведения позиций, риск-менеджмента и другие системы.

Архитектура системы приведена на следующей диаграмме:



В данном документе подробно рассматривается создание клиентов программного интерфейса. Все необходимые для этого функции собраны в библиотеке MTESrl.dll.

Библиотека MTESRL.DLL

Библиотека служит для создания клиентов универсального программного интерфейса, позволяющего подключать к торгово-депозитарным комплексам ММВБ внешние системы распространения торговой информации, сбора клиентских заявок, ведения позиций, риск-менеджмента и другие системы. Библиотека обеспечивает двунаправленную связь с торговой системой и содержит функции как для получения информации из торговой системы (сделки, котировки, инструменты и т.п.), так и для выполнения транзакций (постановка/снятие заявок и т.п.).

В подкаталоге **Demo** каталога установки системы находятся интерфейсные модули к библиотеке и примеры `на Delphi 7 и MS VC 6`.

Сценарий работы с библиотекой

Типичный сценарий работы клиента с программным интерфейсом выглядит так:

1. Регистрация на сервере
2. Получение описания информационных объектов (типов, таблиц и транзакций)
3. Работа с информационными объектами (таблицами и транзакциями)
4. Завершение сеанса связи

РЕГИСТРАЦИЯ НА СЕРВЕРЕ

Для начала работы с интерфейсом необходимо подключиться к серверу MICEX Bridge Server. Для этого служит функция **`MTEConnect`**. Вызывать ее следует до обращения ко всем последующим функциям.

```
function MTEConnect(Params, ErrorMsg: LPSTR): Integer;
```

Аргументы:

Params

Параметры, используемые для установления соединения. Указатель на ASCIIZ-строку, содержащую список параметров, разделенных символами возврата каретки и перевода строки (0x0D, 0x0A) в следующем формате:

```
Parameter1=Value1
Parameter2=Value2
...
ParameterN=ValueN
```

Названия параметров и их допустимые значения зависят от способа соединения конкретной библиотеки с торговой системой. Библиотека MTESRL.DLL использует следующие параметры:

Соединение через интернет:

HOST - список IP-адресов и портов серверов доступа, разделенный запятыми, например: “194.186.240.85:20006,194.186.240.73:20006”;

SERVER - идентификатор сервера доступа, например “EQ_TEST”;

USERID - идентификатор пользователя в торговой системе ММВБ;

PASSWORD - пароль пользователя в торговой системе ММВБ;

INTERFACE - идентификатор интерфейса торговой системы ММВБ, с которым желает работать пользователь;

BOARDS - список режимов, с которыми желает работать пользователь, например: “EQBR,EQNO,EQNL,PSEQ” (необязательный параметр, если не задан будут выбраны все доступные режимы);

Настройка шифрования и ЭЦП «Валидата»

Validata.ProfileName - имя профиля системы криптографической защиты информации «Валидата» (необязательный параметр, если не задан шифрование и ЭЦП будет отключено). Поддерживается также старое имя этого параметра **PROFILENAME**. В качестве имени можно указать зарезервированное слово «My», в этом случае сама «Валидата» предложит выбрать профиль из списка доступных;

Validata.BasePath и **Validata.LdapPath** - альтернативный способ инициализации «Валидаты» (не через имя профиля). Может оказаться полезным, если «Валидата» устанавливалась под другим пользователем (например, клиентская часть шлюза запускается как сервис). В этом случае в реестре текущего пользователя не будет записей об именах профилей, и воспользоваться параметром ProfileName не удастся. Значения этих параметров необходимо уточнить в реестре того пользователя, под которым устанавливалась «Валидата», а именно:

Параметр Validata.BasePath взять из ключа

HKEY_CURRENT_USER\Software\MDPREI\mpki\Profiles\0\BasePath

Параметр Validata.LdapPath взять из ключа

HKEY_CURRENT_USER\Software\MDPREI\mpki\store\2\name

Validata.InitFlags комбинация флагов инициализации «Валидаты»:

1 - не выполнять автоматическое обновление COC (списка отозванных сертификатов) при инициализации;

2 - показывать объекты с истекающим сроком действия при инициализации;

4 - не использовать сетевые справочники;

8 - не выгружать ключ при завершении работы.

Соединение через интерфейс RS-232:

PORT - имя последовательного порта, например, COM1;
BAUDRATE - скорость порта, например, 115200;

Соединение по TCP/IP:

HOST - IP-адрес хоста, на котором работает MICEX Bridge Server;
SERVICE - сервис, на котором работает MICEX Bridge Server;

Соединение по NetBEUI:

HOST - имя хоста, на котором работает MICEX Bridge Server;
PIPE - именованный канал, на котором работает MICEX Bridge Server;

Также во всех случаях поддерживаются параметры:

TIMEOUT - время ожидания выполнения запроса сервером (торговой системой) в мс, например 60000 (1 мин) (значение по умолчанию 30 сек);

SYNCTIME - "0" не синхронизировать время на клиенте с временем сервера (если параметр опущен синхронизация включена), "1" – синхронизировать время;

LOGGING - "0" отключить логирование операций (не создавать log-файл), "1" – включить логирование;

RETRIES - количество попыток восстановить связь с MICEX Bridge Server в случае коммуникационных проблем. (по умолчанию 10);

CONNECTTIME - максимальное время реконнекта, в мс. По умолчанию 1 минута;

LOGFOLDER – каталог для создания лог-файлов работы библиотеки. По умолчанию лог-файлы создаются в одном каталоге с библиотекой.

ErrorMsg

Указатель на буфер размером не менее 256 байт, куда в случае возникновения ошибки будет помещена строка с описанием ошибки.

Возвращаемое значение:

В случае успеха функция возвращает дескриптор установленного соединения. (значение большее или равное MTE_OK) Полученный дескриптор соединения используется в дальнейшем при вызове всех функций MTExxxx.

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. При этом в аргумент ErrorMsg помещается описание проблемы.

Пример:

Установка соединения с сервером через порт COM1 на скорости 115200 бод.

```
Idx: Integer;
ErrorMsg: TErrorMsg;
...
Idx := MTEConnect('PORT=COM1'#13#10'BAUDRATE=115200', @ErrorMsg);
if Idx < MTE_OK then
begin
  Writeln('Ошибка при установке соединения: ' + ErrorMsg);
  Halt;
```

```
end
else
  Writeln('Соединение установлено.');
```

ПОЛУЧЕНИЕ СЛУЖЕБНОЙ И СЕРВИСНОЙ ИНФОРМАЦИИ

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ШЛЮЗЕ

Клиент внешней системы может получать служебную и сервисную информацию о серверной части шлюза ММВБ и соответствующем сервере доступа ММВБ с помощью функции **MTEGetServInfo**.

```
function MTEGetServInfo(Idx: Integer; var ServInfo: PChar; var Len: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, для которого нужно получить информацию.

ServInfo

Указатель на указатель на буфер в который будут помещены данные возвращаемые функцией.

Len

Указатель на переменную в которую будет помещено значение длины данных возвращаемых функцией.

Возвращаемое значение:

В случае успеха функция возвращает MTE_OK и помещает в аргумент ServInfo указатель на буфер следующего формата.

Поле	**Тип данных (IBM PC)**	**Длина поля**	**Описание**
Connected_To_Micex	INTEGER	4	Описывает состояние соединения с ТС ММВБ. Возможное значение: 1 - соединен с ТС ММВБ, 0 -- не соединен.
Session_Id	INTEGER	4	Внутренний номер текущей торговой сессии ММВБ. Изменяется каждую торговую сессию.
MICEX_Sever_Name	CHAR	33	Логическое имя серверов доступа ТС ММВБ, например, GATEWAY, FOND_GATEWAY и т.д. Позволяет определить к какой ТС подсоединен шлюз: корп. рынок, тестовый и т.д.
Version_Major	CHAR	1	Старший номер версии сервера шлюза.
Version_Minor	CHAR	1	Младший номер версии сервера шлюза.
Version_Build	CHAR	1	Номер сборки версии шлюза. Это и предыдущие поля определяют версию шлюза в порядке Major.Minor.Build.
Beta_version	CHAR	1	Является ли данный релиз бета-версией. Возможное значение: если не 0, то это бета и соответственно, это ее номер.
Debug_flag	CHAR	1	Является ли данный релиз отладочной версией. Возможное значение: значение больше 0 указывает на то, что версия является отладочной.
Test_flag	CHAR	1	Является ли данный релиз тестовой

			версией. Возможное значение: значение больше 0 указывает на то, что версия является тестовой.
Start_Time	INTEGER	4	Время старта новой сессии (задается в INI-файле шлюза) в формате ЧЧММСС. Внимание!!! Это целочисленное значение.
Stop_Time_Min	INTEGER	4	Время завершения работы и автоматической остановки шлюза (задается в INI-файле шлюза) в формате ЧЧММСС. Внимание!!! Это целочисленное значение.
Stop_Time_Max	INTEGER	4	Равно Stop_Time_Min.
Next_Event	INTEGER	4	Следующее ожидаемое событие в Диспетчере Расписания для соответствующего интерфейсного сервера. Возможные значения: 0 - ожидается старт новой торговой сессии, 1 - ожидается окончание текущей торговой сессии.
Event_Date	INTEGER	4	Дата ожидаемого события в формате ДДММГГГГ. Внимание!!! Это целочисленное значение.
BoardsSelected	нультерминированная строка символов		Список идентификаторов выбранных режимов торгов, разделенный запятыми.
UserID	CHAR, нультерминированная строка	13	Идентификатор пользователя используемый интерфейсным сервером для данного соединения.
SystemId	CHAR	1	Тип торговой системы, возможные значения: “P” - фондовый рынок ил рынок ГЦБ, “C” - валютный рынок “F” – срочный рынок.
ServerIp	нультерминированная строка символов		IP-адрес сервера доступа торговой системы ММВБ, например, «195.1.3.51».

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx.

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ВЕРСИИ КЛИЕНТСКОЙ БИБЛИОТЕКИ

Клиент внешней системы может получить информацию о версии клиентской библиотеки шлюза с помощью функции **`MTEGetVersion`**.

```
function MTEGetVersion: PChar; stdcall;
```

Аргументы:

отсутствуют

Возвращаемое значение:

Указатель на ASCIIZ-строку, содержащую текстовое описание версии клиентской библиотеки, например, «MTESrl library 3.8.93».

ПОЛУЧЕНИЕ ОПИСАНИЯ ИНФОРМАЦИОННЫХ ОБЪЕКТОВ

Описание информационных объектов торговой системы содержит список таблиц, транзакций, их полей и некоторых вспомогательных объектов, доступных клиенту. Для получения описания используется функция **MTEStructure**.

```
function MTEStructure(Idx: Integer; var Msg: PMTEMsg): Integer;
```

Аргументы:

Idx

Дескриптор соединения, для которого нужно получить информацию.

Msg

Адрес переменной (имеющей тип "указатель на TMTEMsg"), куда будет помещен указатель на буфер, содержащий описание информационных объектов. Формат буфера описан в приложении 1. Структура TMTEMsg определена так:

```
PMTEMsg = ^TMTEMsg;
TMTEMsg = record
  DataLen: Integer;  // Длина следующих далее данных
  Data: record end;  // Данные переменной длины
end;
```

Возвращаемое значение:

В случае успеха функция возвращает MTE_OK и помещает в аргумент Msg указатель на буфер с описанием.

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. Если возвращен код ошибки MTE_TSMR, поле Data структуры Msg содержит текст сообщения об ошибке длиной DataLen символов.

Пример:

Получение описания доступных информационных объектов для сеанса с номером `Idx`.

```
Idx: Integer;       // Инициализирована вызовом MTEConnect
Err: Integer;
Msg: PMTEMsg;
S: string;
...
Err := MTEStructure(Idx, Msg);
if Err <> MTE_OK then
  if Err = MTE_TSMR then
  begin
    SetString(S, @Msg.Data, Msg.DataLen);
    Writeln('Ошибка: ' + S);
  end
  else
    Writeln('Ошибка: ' + MTEErrorMsg(Err))
else
  Writeln('Описание информационных объектов получено.);
```

Структура информационных объектов, возвращаемых данной функцией, может быть получена в виде HTML-файла с помощью команды **Файл|Сохранить структуру** программы TEServer.exe.

РАБОТА С ИНФОРМАЦИОННЫМИ ОБЪЕКТАМИ

Работа с информационными объектами включает работу с таблицами и выполнение транзакций.

ВЫПОЛНЕНИЕ ТРАНЗАКЦИЙ

Активные операции, такие как постановка или снятие заявки, называемые также транзакциями, выполняются с помощью функции **MTEExecTrans**.

```
function MTEExecTrans(Idx: Integer; TransName, Params, ResultMsg: LPSTR):
       Integer;
```

Аргументы:

Idx

Дескриптор соединения, на котором выполняется транзакция.

TransName

Указатель на ASCIIZ-строку с именем транзакции. Допустимые имена могут быть получены вызовом функции MTEStructure.

Params

Указатель на ASCIIZ-строку, содержащую параметры транзакции. Длина строки и ее содержимое должны соответствовать описанию входных полей транзакции, полученном с помощью MTEStructure. Все поля должны быть представлены в текстовом виде в формате торговой системы следующим образом:

Char Дополняется справа пробелами до длины, указанной в описании поля. Например, для поля типа Char(12) строка "ROOT" должна быть представлена как "ROOT "

Integer Дополняется слева нулями до нужной длины. Например, значение 127 с типом Integer(10) преобразуется в строку "0000000127"

Fixed Оставляется два знака после десятичной точки, убирается десятичная точка, дополняется слева нулями до нужной длины. Например, значение 927,4 с типом Fixed(8) преобразуется в строку "00092740"

Float Оставляется N знаков после десятичной точки, убирается десятичная точка, дополняется слева нулями до нужной длины. Значение N зависит от формата представления цен для финансового инструмента, к которому относится данное поле. Например, значение 26,75 с типом Float(9) для инструмента с N = 4 преобразуется в строку "000267500"

Date Представляется в формате YYYYMMDD. Например значение 24 августа 1999г. преобразуется к "19990824"

Time Представляется в формате HHMMSS. Например значение 16:27:39 преобразуется к "162739"

ResultMsg

Указатель на буфер размером не менее 256 байт, куда в случае успешного выполнения будет помещена строка текста с результатом обработки транзакции торговой системой.

Возвращаемое значение:

Если транзакция была обработана торговой системой, возвращается следующее:

MTE_OK - транзакция выполнена;
MTE_TRANSREJECTED - транзакция обработана, но была отвергнута торговым сервером (недопустимые параметры, нет прав на выполнение и т.п.);
MTE_TSMR - фатальный сбой при выполнении транзакции (потеря соединения с торговой системой и т.п.).

При этом в аргумент *ResultMsg* помещается строка текста с результатом обработки транзакции торговой системой.

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. Значение поля *ResultMsg* при этом не определено.

Пример:

Допустим в описании информационных объектов, полученном с помощью MTEStructure, определена транзакция "Поставить заявку" со следующими полями:

```
ORDER                       // Имя транзакции
  BuySell: Char(1)          // "B" - покупка, "S" - продажа
  SecCode: Char(17)         // код инструмента
  Price: Float(9)           // цена
  Quantity: Integer(10)     // кол-во лотов
```

Приведенный ниже фрагмент кода ставит заявку на покупку 14 лотов инструмента "0CURRUSD000000TOD" по цене 26,15 (для этого инструмента формат представления цен содержит 4 знака после десятичной точки):

```
Idx: Integer;       // Инициализирована вызовом MTEConnect
Err: Integer;
ResultMsg: TErrorMsg;
...
Err := TEExecTrans(Idx, 'ORDER',
       'B0CURRUSD000000TOD00026150000000000014', @ResultMsg);
case Err of
  MTE_OK: Writeln('Транзакция выполнена: ' + ResultMsg);
  MTE_TSMR, MTE_TRANSREJECTED: Writeln('Транзакция НЕ выполнена: ' +
       ResultMsg);
  else Writeln('Ошибка: ' + MTEErrorMsg(Err));
end;
```

Примечание: Для каждого соединения, транзакции и информационные запросы передаются в торговую систему последовательно: отправка новой транзакции или запроса возможна только после получения ответа на предыдущий запрос. Для исключения задержек, рекомендуется:

- Использовать раздельные соединения для передачи транзакций и для запросов данных таблиц торговой системы.
- При больших пиковых частотах транзакций, использовать несколько соединений с балансировкой их загрузки.

РАБОТА С ТАБЛИЦАМИ

Работа с таблицами включает в себя следующие шаги:

1. Открытие таблицы
2. Периодический запрос изменений
3. Закрытие таблицы

ОТКРЫТИЕ ТАБЛИЦЫ

Работа с таблицей торговой системы начинается с вызова функции **`MTEOpenTable`**. Функция открывает таблицу и возвращает часть или все текущее содержимое таблицы.

```
function MTEOpenTable(Idx: Integer; TableName, Params: LPSTR;
       Complete: BOOL; var Msg: PMTEMsg): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова `MTEConnect`.

TableName

Указатель на ASCIIZ-строку с именем таблицы. Допустимые имена могут быть получены вызовом функции MTEStructure.

Params

Указатель на ASCIIZ-строку, содержащую параметры таблицы. Длина строки и ее содержимое должны соответствовать описанию входных полей таблицы, полученном с помощью MTEStructure. Все поля должны быть представлены в текстовом виде в формате торговой системы (см. MTEExecTrans).

Complete

Флаг, позволяющий запросить все содержимое таблицы или только часть. Используется следующим образом:

TRUE Функция возвращает всю информацию, содержащуюся в данный момент в таблице. Выполняет столько обращений к торговой системе, сколько нужно для получения всех данных. При большом объеме таблицы (например "Сделки") может выполняться долго. Если все содержимое сразу не требуется и чтобы уменьшить время выполнения, следует использовать значение FALSE.

FALSE Функция возвращается только часть данных или вообще ничего, в зависимости от типа таблицы. Выполняет не более одного обращения к торговой системе. Остальные данные рассматриваются как обновления и должны дочитываться в цикле запроса изменений с помощью вызовов MTEAddTable/MTERefresh.

Msg

Адрес переменной (имеющей тип "указатель на TMTEMsg"), куда в случае успеха будет помещен указатель на буфер, содержащий информацию из открытой таблицы. Формат буфера описан в приложении 2.

Возвращаемое значение:

В случае успеха функция возвращает дескриптор открытой таблицы (значение большее или равное MTE_OK) Полученный дескриптор используется в дальнейшем при вызове функции MTEAddTable.

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. Если возвращен код ошибки MTE_TSMR, поле Data структуры Msg содержит текст сообщения об ошибке длиной DataLen символов.

ЗАПРОС ИЗМЕНЕНИЙ

Запрос изменений выполняется в пакетном режиме, т.е. одновременно запрашиваются изменения по нескольким открытым таблицам. Для этого запрос сначала формируется путем нескольких вызовов **MTEAddTable**, а затем выполняется с помощью **MTERefresh**. Вызов других функций библиотеки (кроме MTEErrorMsg) между двумя этими функциями запрещен.

Функция MTEAddTable добавляет в очередь (пакет запросов) запрос на получение изменений в таблице, с момента предыдущего запроса изменений.

```
function MTEAddTable(Idx, HTable, Ref: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова MTEConnect.

HTable

Дескриптор таблицы, полученный с помощью вызова MTEOpenTable.

Ref

Дополнительный параметр, используемый клиентом по своему усмотрению. Применяется обычно для идентификации информации, предназначенной данной таблице, в буфере, возвращаемом функцией MTERefresh.

Возвращаемое значение:

Один из кодов ошибки MTE_xxxx.

Функция MTERefresh отправляет на сервер пакет запросов на получение изменений, сформированный вызовами MTEAddTable, и возвращает эти изменения.

```
function MTERefresh(Idx: Integer; var Msg: PMTEMsg): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова MTEConnect.

Msg

Адрес переменной (имеющей тип "указатель на TMTEMsg"), куда в случае успеха будет помещен указатель на буфер, содержащий полученные обновления. Формат буфера описан в приложении 3.

Возвращаемое значение:

В случае успеха функция возвращает MTE_OK и помещает в аргумент Msg указатель на полученные данные

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. Если возвращен код ошибки MTE_TSMR, поле Data структуры Msg содержит текст сообщения об ошибке длиной DataLen символов.

ЗАКРЫТИЕ ТАБЛИЦЫ

По окончании работы с таблицей ее необходимо закрыть используя функцию **MTECloseTable**. После вызова этой функции дескриптор таблицы не может более использоваться.

```
function MTECloseTable(Idx, HTable: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова MTEConnect.

HTable

Дескриптор закрываемой таблицы, полученный с помощью вызова MTEOpenTable.

Возвращаемое значение:

Один из кодов ошибки MTE_xxxx.

ПРИМЕР РАБОТЫ С ТАБЛИЦАМИ

Допустим в описании информационных объектов, полученном с помощью MTEStructure, определены таблицы "Ценные бумаги" и "Сделки" со следующими входными полями:

```
SECURITIES                // Имя таблицы (Ценные бумаги)
  Market: Char(4)         // Код рынка
  Board: Char(4)          // Код режима торгов

TRADES                    // Таблицы "Сделки" без параметров
```

В следующем фрагменте кода показано как следует работать с таблицами торговой системы. Таблицы открываются, периодически запрашивается изменение их содержимого, и затем таблицы закрываются.

```
Idx: Integer;       // Инициализирована вызовом MTEConnect
Msg: PMTEMsg;
HSecurs, HTrades: Integer;
...

HSecurs := MTEOpenTable(Idx, 'SECURITIES', 'CURR    ', True, Msg);
...
// Обработка полученных данных
...
HTrades := MTEOpenTable(Idx, 'TRADES', '', False, Msg);
...
// Обработка полученных данных
...

repeat
  MTEAddTable(Idx, HSecurs, 0);
  MTEAddTable(Idx, HTrades, 1);
  MTERefresh(Idx, Msg);
  ...
  // Обработка обновлений
  ...
until Terminated;

MTECloseTable(Idx, HSecurs);
MTECloseTable(Idx, HTrades);
```

ЗАМЕЧАНИЯ ПО РАБОТЕ С ТАБЛИЦАМИ

Замечание 1. Во избежание разрыва соединения по тайм-ауту со стороны сервера рекомендуется: во-первых, в настройке шлюза teap.ini не устанавливать слишком маленькое (меньше 60 секунд) значение DisconnectIfIdleFor; во-вторых, регулярно (примерно с интервалом в 30 секунд) поддерживать соединение в активном состоянии – например, запрашивая обновление таблицы TESYSTIME.

Замечание 2. Большинство таблиц торговой системы могут быть открыты и закрыты в произвольный момент времени внутри сеанса связи с сервером, произвольное количество раз, можно открыть произвольное число экземпляров одной и той же таблицы. Однако из-за ограничений, накладываемых реализацией торговой системы, некоторые таблицы могут быть открыты только один раз в течение сеанса. К таким таблицам относится, например, таблица ORDERS ("Заявки"). Если такую таблицу закрыть, а затем открыть вновь, то содержимое таблицы, полученное в первый раз, вновь получено не будет, а будут приходить только изменения.

В связи с вышесказанным, такие таблицы рекомендуется открывать только один раз в течение сеанса связи и закрывать их только при завершении сеанса.

Замечание 3. Для таблиц с установленным флагом "tfClearOnUpdate - Очищать при обновлении" (кроме таблицы ORDERBOOK) определен следующий порядок обработки обновлений: когда таблица должна быть полностью очищена КолвоСтрок устанавливается равным 1, то есть, возвращается одна строка со значением ДлинаДанных = 0 (см. приложение 2).

Для таблицы ORDERBOOK существуют два режима запроса информации по котировкам:

1. для получения информации по одному инструменту, в запросе задаются непустые значения для полей "Режим" и "Инструмент";
2. для получения информации по котировкам всех доступных инструментов в одном запросе, поля "Режим" и "Инструмент" заполняются символами пробела.

Соответственно, для первого способа в случае, когда таблица котировок должна быть очищена в ответ на запрос, приходит таблица с одной строкой, содержащей следующие значения: КолвоПолей =2 и ДлинаДанных = (длина поля "Режим" + длина поля "Инструмент") В этой строке содержатся только поля "Режим" и "Инструмент". Для второго случая в ответе на запрос могут содержаться несколько таких строк (в которых присутствуют только значения полей "Режим" и "Инструмент"), что для данных инструментов означает очистку значений котировок.

Обратите внимание на еще два момента: при первом запросе таблицы для всех доступных инструментов, т.е. при открытии, могут прийти строки изначально с нулевыми котировками. Это связано с логикой выдачи информации Торговой Системой: по данным бумагам происходили какие-либо изменения в статусе и ТС выдает данные изменения в котировочных полях, которые не отображаются в клиентских системах. Поэтому и происходит выдача всех измененных котировок, даже если они пустые. При последующих запросах выдается информация по только измененным котировкам.

Второй момент: TEClient.exe отображает в окне котировок для всех инструментам только данные последнего запроса на изменения, т.е. только те котировки, в которых произошли изменения.

Замечание 4. Рекомендуемая частота запросов обновлений - 1 раз в секунду. Для исключения задержек в получении информации при пиках активности рынка допускается адаптивный интервал между запросами:
Если объем полученного пакета данных превышает 30 кбайт, немедленно отправить новый запрос на обновление. Если объем данных в пакете не превышает этой величины, следующий запрос может быть отправлен через обычный интервал времени (1 секунду).

Оптимизация использования памяти

Все функции библиотеки MTESrl.dll, возвращающие указатель на буфер с информацией (указатель на структуру `PMTEMsg`, например, `MTEStructure`, `MTERefresh` и другие) используют в качестве приемного буфера одну и ту же область памяти (в рамках одного соединения, для разных соединений используются разные области памяти). Назовем такие функции информационными.

Если при очередном вызове информационной функции размер получаемых данных превышает размер выделенного для приема буфера, происходит выделение (Reallocation) большего блока памяти. Таким образом максимальный размер выделенной памяти равен размеру максимального полученного блока информации. Вся выделенная память освобождается при завершении соединения с помощью функции `MTEDisconnect`.

Существует возможность освободить память, используемую в качестве приемного буфера, в произвольный момент времени, не завершая соединения. Для этого предназначена функция `MTEFreeBuffer`. Эту функцию следует вызывать только после обработки всех принятых данных. Следует помнить, что это приведет к необходимости выделения памяти при следующем вызове одной из информационных функций. Частый вызов функции `MTEFreeBuffer` может отрицательно повлиять на производительность.

```
function MTEFreeBuffer(Idx: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова `MTEConnect`, которое надо закрыть.

Возвращаемое значение:

Один из кодов ошибки `MTE_xxxx`.

Восстановление после сбоя на MICEX Bridge Server

Библиотека MTESRL.DLL позволяет начать получение данных от MICEX Bridge Server не с “нуля”, а с некоторого момента. Для этого предварительно должен быть сохранен “снимок”

состояния открытых таблиц. Впоследствии, например, в случае потери соединения с MICEX Bridge Server, можно восстановить состояние открытых таблиц и продолжить получение информации. Ниже приведен подробный сценарий работы в таких случаях.

Для получения текущего состояния открытых на сервере таблиц используется следующая функция:

```
function MTEGetSnapshot(Idx: Integer; var Snapshot: PChar;
       var Len: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, для которого необходимо получить «снимок» открытых таблиц.

Snapshot

Адрес переменной, куда в случае успеха будет помещен указатель на «снимок».

Len

Адрес переменной, куда в случае успеха будет помещена длина «снимка» (буфера, указатель на который находится в *Snapshot*).

Возвращаемое значение:

В случае успеха функция возвращает `MTE_OK`.

При возникновении ошибки возвращается один из кодов ошибки `MTE_xxxx`. Если возвращен код ошибки `MTE_TSMR`, аргумент *Snapshot* указывает на текст сообщения об ошибке, а аргумент *Len* содержит длину этого сообщения.

«Снимок» открытых на сервере таблиц может рассматриваться просто как буфер некоторых двоичных данных. Его содержимое не несет для клиента никакой смысловой нагрузки.

Состояние открытых на сервере таблиц может быть в любой момент времени восстановлено по сделанному ранее “снимку”.

```
function MTESetSnapshot(Idx: Integer; Snapshot: PChar; Len: Integer;
       ErrorMsg: LPSTR): Integer;
```

Аргументы:

Idx

Дескриптор соединения, для которого восстанавливается состояние.

Snapshot

Указатель на буфер, содержащий предварительно снятый «снимок».

Len

Длина буфера, на который указывает Snapshot.

ErrorMsg

Указатель на буфер размером не менее 256 байт, куда будет помещена строка текста с результатом восстановления состояния.

Возвращаемое значение:

Если функция была обработана торговой системой, возвращается следующее:

MTE_OK – восстановление выполнено;
MTE_TSMR - торговая система не смогла восстановить состояние.

При этом в аргумент *ErrorMsg* помещается строка текста с результатом, возвращенным торговой системой.

При возникновении ошибки возвращается один из кодов ошибки MTE_xxxx. Значение поля *ErrorMsg* при этом не определено.

Пример восстановления после сбоя

Предположим, что мы:

1. установили соединение с MICEX Bridge Server с помощью **MTEConnect**;
2. открыли несколько таблиц с помощью **MTEOpenTable** и сохранили их дескрипторы в переменных hTable1, hTable2, ..., hTableN;
3. выполняли транзакции и запрашивали обновления информационных таблиц, периодически сохраняя «снимок» состояния с помощью функции **MTEGetSnapshot**;

Допустим, в какой-то момент соединение с MICEX Bridge Server было нарушено. Процедура восстановления будет выглядеть так:

1. Заново устанавливаем соединение с MICEX Bridge Server с помощью **MTEConnect**;
2. Вызываем **MTESetSnapshot** с последним сохраненным «снимком»
3. Теперь можем пользоваться старыми дескрипторами таблиц hTable1, hTable2, ..., hTableN, открытыми в предыдущем сеансе. Вызывать **MTEOpenTable** не нужно. Последующие вызовы функции MTERefresh будут возвращать обновления таблиц, накопленные после сохранения Snapshot.

Если данные, полученные до обрыва связи были сохранены, использование механизма Get/Set Snapshot позволяет существенно уменьшить время получения всех обновлений таблиц после восстановления соединения.

Завершение сеанса связи

По окончании работы с рынком клиент должен вызвать функцию **MTEDisconnect**.

```
function MTEDisconnect(Idx: Integer): Integer;
```

Аргументы:

Idx

Дескриптор соединения, полученный с помощью вызова `MTEConnect`, которое надо закрыть.

Возвращаемое значение:

Один из кодов ошибки `MTE_xxxx`.

Пример:

Закрываем соединение, имеющее дескриптор Idx.

```
Idx: Integer;       // Инициализирована вызовом MTEConnect
Err: Integer;
...
Err := MTEDisconnect(Idx);
if Err <> MTE_OK then Writeln(MTEErrorMsg(Err)
  else Writeln('Сеанс работы с рынком завершен');
```

Сообщения об ошибках

Все функции библиотеки возвращают коды ошибок MTE_xxxx. Для получения текстового описания по коду ошибки может использоваться функция **MTEErrorMsg**.

```
function MTEErrorMsg(ErrCode: Integer): LPSTR;
```

Аргументы:

ErrorCode

Один из кодов MTE_xxxx.

Возвращаемое значение:

Указатель на ASCIIZ-строку, содержащую текстовое описание ошибки.

Коды ошибок

ID	Код	Описание
MTE_OK	0	Нет ошибок.
MTE_CONFIG	-1	Ошибка в конфигурации: не удается открыть COM-порт (для TEServer), подключение к неправильному серверу, на шлюзовом сервере не указаны сервисы, неверные значения параметров в конфигурационном файле.
MTE_SRVUNAVAIL	-2	Сервер не доступен. Не запущен MICEX Bridge Server, недоступна торговая система, либо нарушена связь.
MTE_LOGERROR	-3	При вызове MTEConnect не удалось создать log-файл.
MTE_INVALIDCONNECT	-4	Задан недопустимый дескриптор соединения. Не было вызова MTEConnect, либо уже был вызвана функция MTEDisconnect.
MTE_NOTCONNECTED	-5	Соединение с указанным дескриптором было разорвано вследствие возникновения ошибки (не в результате вызова MTEDisconnect). Ошибка в MICEX Bridge Server, торговая система завершила работу, либо нарушена связь.
MTE_WRITE	-6	Ошибка записи в порт. Ошибка в MICEX Bridge Server, либо нарушена связь.
MTE_READ	-7	Ошибка чтения из порта. Ошибка в MICEX Bridge Server, либо нарушена связь.
MTE_TSMR	-8	Ошибка на уровне протокола взаимодействия с торговой системой ММВБ, либо торговая система недоступна.
MTE_NOMEMORY	-9	Недостаточно памяти для выполнения операции.
MTE_ZLIB	-10	Ошибка при сжатии/распаковке передаваемых данных.
MTE_PKTINPROGRESS	-11	Была вызвана функция MTEAddTable без последующего вызова MTERefresh. Во время сборки пакета запросов на обновление вызов других функций библиотеки невозможен.
MTE_PKTNOTSTARTED	-12	Была вызвана функция MTERefresh без предварительного вызова MTEAddTable. Сначала необходимо сформировать пакет запросов на обновление.
MTE_FATALERROR	-13	Непредвиденная критическая ошибка.
MTE_INVALIDHANDLE	-14	Неверный дескриптор таблицы. Дескриптор не был получен вызовом MTEOpenTable, либо таблица уже закрыта с помощью MTECloseTable.
MTE_DSROFF	-15	Связь по последовательному порту нарушена (отсутствует сигнал DSR). Возможно нарушена целостность последовательного кабеля, либо последовательный порт закрыт на одной из сторон соединения.
MTE_UNKNOWN	-16	Во время выполнения функции произошла непредвиденная ошибка.
MTE_BADPTR	-17	Одной из функций MTExxxx() в качестве аргумента передан недопустимый указатель.
MTE_TRANSREJECTED	-18	Торговая система ММВБ обработала запрос и вернула код ошибки. Транзакция не выполнена.
MTE_TEUNAVAIL	-19	Торговая система ММВБ временно недоступна. Сервер продолжает попытки подключения к ТС или находится в

		ожидании торговой сессии.
MTE_NOTLOGGEDIN	-20	Клиент пытается отправить запрос через сервер после того, как тот установил новое подключение к Торговой системе. Требуется заново подключить клиента к серверу.
MTE_WRONGVERSION	-21	Сервер не поддерживает эту версию клиентской библиотеки.
MTE_LOGON	-30	При соединении с сервером были указаны неверные регистрационные параметры: USERID, PASSWORD и т.п.
MTE_TOOSLOWCONNECT	-31	Слишком медленный канал связи не дает корректно завершить процедуру коннекта/реконнекта.
MTE_CRYPTO_ERROR	-32	Ошибка при шифровании/расшифровке, создании/проверке ЭЦП.
MTE_THREAD_ERROR	-33	Клиент пытается использовать одно соединение в двух потоках. Например, пытается вызвать функцию MTExxxx() в то время, как ещё не завершил работу предыдущий вызов MTExxxx().
MTE_NOTIMPLEMENTED	-34	Вызываемая функция отсутствует в данной версии клиентской библиотеки.

ПРИЛОЖЕНИЕ 1. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTESTRUCTURE

Поле Data структуры TMTEMsg, указатель на которую возвращает функция MTEStructure, имеет следующий формат (описание элементарных типов String, Integer и т.п. см. прил. 4; в случае структуры, возвращаемой функцией MTEStructure, перед каждым полем типа String передаётся 4 байта, содержащие длину этой строки):

```
        поле                    тип
TInterface:
        ИмяИнтерфейса           String
        ОписаниеИнтерфейса      String
        ПеречислимыеТипы        TEnumTypes
        Таблицы                 TTables
        Транзакции              TTransactions
```

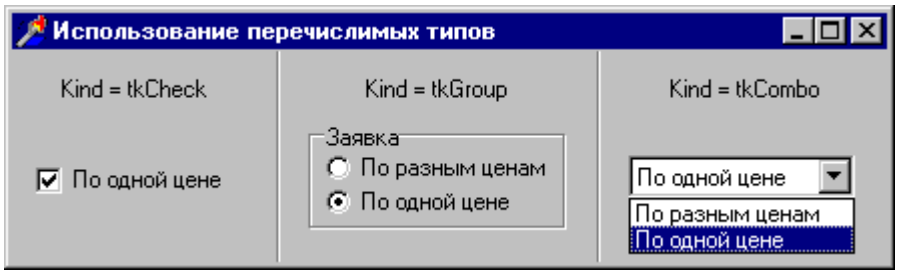
Описание информационных объектов состоит из трех блоков: описание перечислимых типов, таблиц и транзакций.

```
TEnumTypes:
        КолвоТипов              Integer
        Тип1                    TEnumType
        Тип2                    TEnumType
        ...
        ТипN                    TEnumType

TEnumType:
        Имя                     String
        Описание                String
        Размер                  Integer
        Тип                     TEnumKind
        КолвоКонстант           Integer
        Константа1              String
        Константа2              String
        ...
        КонстантаN              String

TEnumKind:                      Integer
        ekCheck = 0
        ekGroup = 1
        ekCombo = 2
```

Перечислимые типы используются для описания допустимых значений полей таблиц и транзакций. Описание типа может выглядеть например так:

```
'TCurrency'          // Имя
'Валюта'             // Описание
4                    // Размер
ekCombo              // Предпочтительный вид представления - "Тип"
3                    // Кол-во констант
'RUR =Рубли'         // Константа 1
'USD =Доллары'       // Константа 2
'DEM =Марки'         // Константа 3
```

Поле "Размер" (=4) указывает размер допустимых значений для полей, имеющих данный тип.
Поле "Тип" (=ekCombo) задает предпочтительный способ представления поля, используемый при создании формы ввода параметров. Например, поле с типом ekCombo может быть представлено в виде списка значений. Возможные варианты показаны на следующем рисунке:



Константы состоят из двух частей - допустимого значения (всегда длиной "Размер") и описания этого значения, разделенных символом равенства (=).

```
TTables:
       КолвоТаблиц          Integer
       Таблица1             TTable
       Таблица2             TTable
       ...
       ТаблицаN             TTable

TTable:
       Имя                  String
       Описание             String
       Атрибуты             TTableFlags
       ВходныеПоля          TFields
       ВыходныеПоля         TFields

TTableFlags:                Integer
       tfUpdateable    = 1
       tfClearOnUpdate = 2
```

Список входных полей таблицы используется при формировании строки параметров для функции `MTEOpenTable`.
Список выходных параметров позволяет разбирать буфера, возвращаемые функциями `MTEOpenTable` и `MTERefresh`.
Атрибуты таблицы могут комбинироваться (то есть значение будет равно 3) и имеют следующие значения:

tfUpdateable - таблица является обновляемой. Для нее можно вызывать функции MTEAddTable/MTERefresh;
tfClearOnUpdate - старое содержимое таблицы должно удаляться при получении каждого обновления с помощью функций MTEAddTable/MTERefresh.

```
TFields:
       КолвоПолей           Integer
       Поле1                TField
       Поле2                TField
```

```
       ...
       Поле_N                TField

TField:
       Имя                   String
       Описание              String
       Размер                Integer
       Тип                   TFieldType
       Атрибуты              TFieldFlags
       ПеречислимыйТип       String
       ЗначениеПоУмолчанию   String

TFieldType:                  Integer
       ftChar    = 0
       ftInteger = 1
       ftFixed   = 2
       ftFloat   = 3
       ftDate    = 4
       ftTime    = 5

TFieldFlags:                 Integer
       ffKey       = 0x01
       ffSecCode   = 0x02
       ffNotNull   = 0x04
       ffVarBlock  = 0x08
       ffFixed1    = 0x10
       ffFixed3    = 0x20
       ffFixed4    = 0x30
```

Атрибуты поля (TFieldFlags) могут комбинироваться и имеют следующие значения:

ffKey	Поле является ключевым. Строки таблицы с совпадающими значениями ключевых полей должны объединяться в одну строку.
ffSecCode	Поле содержит код финансового инструмента.
ffNotNull	Поле не может быть пустым.
ffVarBlock	Поле входит в группу полей, которые могут повторяться несколько раз.
ffFixed1 *ffFixed3* *ffFixed4*	Модификатор для полей с типом ftFixed: наличие флагов ffFixed1, ffFixed3 или ffFixed4 означает, что число десятичных знаков после запятой равно 1, 3 или 4, соответственно, вместо используемого по умолчанию значения 2.

Примечание. В списке выходных полей таблицы, в отличие от входных, отсутствует поле "ЗначениеПоУмолчанию".
"Размер" задает длину поля в символах.
"ПеречислимыйТип" может содержать имя перечислимого типа, к которому относится поле, или пустую строку.
"Значение по умолчанию" может использоваться при создании формы ввода параметров.
Все поля представлены в текстовом виде в формате торговой системы (см. `MTEExecTrans`).

```
TTransactions:
       КолвоТранзакций       Integer
       Транзакция_1          TTransaction
       Транзакция_2          TTransaction
       ...
       Транзакция_N          TTransaction

TTransaction:
       Имя                   String
       Описание              String
       ВходныеПоля           TFields
```

Список входных полей транзакции используется при формировании строки параметров для функции `MTEExecTrans`.

ПРИЛОЖЕНИЕ 2. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTEOPENTABLE

Поле Data структуры TMTEMsg, указатель на которую возвращает функция MTEOpenTable, содержит строки запрошенной таблицы и имеет следующий формат (описание элементарных типов String, Integer и т.п. см. прил. 4):

поле	тип
TMTETable:	
Ref	Integer
КолвоСтрок	Integer
$Строка_1$	TMTERow
$Строка_2$	TMTERow
...	
$Строка_N$	TMTERow

Поле "Ref" используется при запросе изменений сразу по нескольким таблицам с помощью функций MTEAddTable/MTERefresh. Оно содержит значение, переданное в качестве третьего параметра функции MTEAddTable(Idx, HTable, Ref). По значению этого поля можно определить, какой таблице (дескриптор HTable) соответствует полученная структура TMTETable. В буфере, возвращаемом MTEOpenTable, значение поля "Ref" зарезервировано и равно 0.

TMTERow:	
КолвоПолей	Byte
ДлинаДанных	Integer
НомераПолей	Byte[КолвоПолей]
ДанныеПолей	Byte[ДлинаДанных]

Строки таблицы имеют переменную длину и могут содержать разное число полей.

Поле "КолвоПолей" содержит число полей таблицы, присутствующих в данной строке. Если значение это поля равно 0, в строке присутствуют все поля таблицы (см MTEStructure).

Поле "ДлинаДанных" содержит суммарный размер полей таблицы в данной строке.

Поле "НомераПолей" имеет переменную длину. Его размер равен значению поля "КолвоПолей". Поле содержит номера полей (по одному байту на номер), присутствующих в данной строке. Номер поля соответствует порядковому номеру выходного поля в описании информационных объектов (см MTEStructure). Если "КолвоПолей" равно 0, значит "НомераПолей" отсутствует, а в качестве номеров полей следует брать последовательность номеров 0, 1, 2, 3 … N.

Поле "ДанныеПолей" (размером "ДлинаДанных" байт) содержит набор значения полей таблицы. Количество полей определяются значением "КолвоПолей", а их суммарная длина - "ДлинаДанных". Длина и тип каждого конкретного поля определяются в описании информационных объектов (см MTEStructure). Все поля представлены в текстовом виде в формате торговой системы (см. `MTEExecTrans`).

Пример:

Допустим в описании информационных объектов, полученном с помощью MTEStructure, определена таблица "Сделки" со следующими выходными полями:

```
TRADES                    // "Сделки"
  TradeNum: Integer(12)   // Номер сделки
  TradeTime: Char(6)      // Время сделки
  BuySell: Char(1)        // "B" - покупка, "S" - продажа
  SecCode: Char(17)       // код инструмента
  Price: Float(9)         // цена
  Qty: Integer(10)        // кол-во лотов
```

Вызвана функция:

```
MTEOpenTable(Idx, 'TRADES', '', True, Msg);
```

В результате в поле Msg.Data содержится следующая информация

```
{
  0x00000000,              // Поле "Ref"
  0x00000002,              // Получено 2 строки
  0x04,                    // В первой строке 4 поля
  0x00000030,              // Длина данных 48 байт
  #0#3#4#5,                // Номера полей 0, 3, 4, 5:
// это поля "TradeNum", "SecCode", "Price", "Qty" из описания
  '0000001205670CURRUSD000000TOD0002579000000000037'
// Значения полей: 120567, "0CURRUSD000000TOD", 25.79, 37
  0x02,                    // Во второй строке 2 поля
  0x17,                    // Длина данных 23 байта
  #1#3,                    // Номера полей 1, 3:
// это поля "TradeTime" и "SecCode" из описания
  '1029530CURRUSD000000TOM'
// Значения полей: "10:29:53" и "0CURRUSD000000TOM"
}
```

ПРИЛОЖЕНИЕ 3. ФОРМАТ БУФЕРА ДЛЯ ФУНКЦИИ MTEREFRESH

Поле Data структуры TMTEMsg, указатель на которую возвращает функция MTEOpenRefresh, содержит несколько таблиц торговой системы и имеет следующий формат (описание элементарных типов String, Integer и т.п. см. прил. 4):

поле	тип
TMTETables:	
КолвоТаблиц	Integer
$Таблица_1$	TMTETable
$Таблица_2$	TMTETable
...	
$Таблица_N$	TMTETable

Таким образом, буфер содержит несколько таблиц. Формат буфера таблицы описан в приложении 2.

ПРИЛОЖЕНИЕ 4. ЭЛЕМЕНТАРНЫЕ ТИПЫ

Для представления элементарных типов в библиотеке MTESRL.DLL используются следующие структуры:

Byte

Один байт.

Integer

Четыре байта в формате процессоров x86 (сначала наименее значащий байт).

String

Структура следующего вида:

```
ДлинаСтроки: Integer
ТекстСтроки: Byte[ДлинаСтроки]
```

Byte[N]

Массив байт длиной N.

Приложение 5. Форматирование полей с типами FLOAT, FIXED, DATE, TIME

Значения полей типа Float (вещественные числа) передаются в текстовом представлении без десятичной точки. Количество знаков после десятичной точки в полях типа Float для конкретной ценной бумаги определяется значением поля "DECIMALS" таблицы "SECURITIES".

В полях типа Float обязательно должны присутствовать DECIMALS знаков после запятой. Например, число 465,39 для ценной бумаги с DECIMALS = 4 должно быть представлено как "4653900". Значение "46539" в этом случае будет воспринято торговой системой как 4,6539.

В полях типа Fixed значения также передаются в текстовом представлении без десятичной точки. По умолчанию поля данного типа имеют два знака после десятичной точки. Однако при наличии атрибутов поля ffFixed1, ffFixed3, ffFixed4 (см. Приложение 1), число десятичных знаков равно 1, 3 или 4, соответственно.

Значения в полях типа Date передаются в виде текстовой строки ДДММГГГГ.

Значения в полях типа Time передаются в виде текстовой строки формата ЧЧММСС.