



# Интерпретатор языка Lua

---

Руководство пользователя

Версия 2.3

The logo for Quik, consisting of the word "quik" in a white, lowercase, sans-serif font, centered within a solid green circle.

quik

# Содержание

|   |    |
|---|----|
| 1. О программе .....  | 5  |
| 1.1 Возможности .....   | 5  |
| 1.2 Подготовка к работе .....   | 5  |
| 1.3 Работа с программой .....   | 5  |
| 2. Функции терминала, доступные в скрипте Lua .....                       | 7  |
| 2.1 Сервисные функции .....   | 7  |
| 2.2 Функции обратного вызова .....  | 11 |
| 3. Функции взаимодействия скрипта Lua и Рабочего места QUIK .....         | 19 |
| 3.1 Функции для обращения к строкам произвольных таблиц QUIK .....        | 19 |
| 3.2 Функции для обращения к спискам доступных параметров .....            | 22 |
| 3.3 Функция для получения информации по денежным средствам .....          | 23 |
| 3.4 Функция для получения информации по бумажным лимитам .....            | 24 |
| 3.5 Функция для получения информации по фьючерсным лимитам .....          | 25 |
| 3.6 Функция для получения информации по фьючерсным позициям .....         | 25 |
| 3.7 Функция для получения информации по инструменту .....                 | 25 |
| 3.8 Функция для получения даты торговой сессии .....                      | 26 |
| 3.9 Функция для получения стакана по указанному классу и бумаге .....     | 26 |
| 3.10 Функции для работы с графиками .....                                 | 27 |
| 3.11 Функции для работы с заявками .....                                  | 31 |
| 3.12 Функции для получения значений Таблицы текущих торгов .....          | 33 |
| 3.13 Функции для получения параметров таблицы «Клиентский портфель» ..... | 34 |
| 3.14 Функции для получения параметров таблицы «Купить/Продать» .....      | 38 |
| 3.15 Функции для работы с таблицами Рабочего места QUIK .....             | 41 |
| 3.16 Функции для работы с метками .....                                   | 49 |
| 3.17 Функции для заказа стакана котировок .....                           | 51 |
| 3.18 Функции для заказа параметров Таблицы текущих торгов .....           | 52 |
| 4. Структуры данных .....   | 53 |

|      |   |    |
|------|---|----|
| 4.1  | Классы .....  | 53 |
| 4.2  | Фирмы .....   | 54 |
| 4.3  | Обезличенные сделки.....  | 54 |
| 4.4  | Сделки.....   | 55 |
| 4.5  | Заявки.....   | 58 |
| 4.6  | Текущие позиции по клиентским счетам .....  | 60 |
| 4.7  | Текущие позиции по бумагам.....   | 61 |
| 4.8  | Стоп-заявки.....  | 61 |
| 4.9  | Лимиты по фьючерсам .....   | 63 |
| 4.10 | Позиции по клиентским счетам (фьючерсы).....  | 64 |
| 4.11 | Лимиты по денежным средствам .....  | 65 |
| 4.12 | Удаление денежного лимита.....  | 66 |
| 4.13 | Удаление бумажного лимита.....  | 66 |
| 4.14 | Удаление фьючерсного лимита .....   | 67 |
| 4.15 | Лимиты по бумагам.....  | 67 |
| 4.16 | Денежные позиции .....  | 68 |
| 4.17 | Заявки на внебиржевые сделки .....  | 68 |
| 4.18 | Сделки для исполнения.....  | 71 |
| 4.19 | Торговые счета.....   | 73 |
| 4.20 | Отчеты по сделкам для исполнения .....  | 74 |
| 4.21 | Инструменты.....  | 75 |
| 4.22 | Свечки графика .....  | 76 |
| 4.23 | Формат даты и времени, используемый в таблицах .....                                      | 76 |
| 4.24 | Транзакции.....   | 77 |
| 5.   | Описание битовых флагов.....  | 78 |
| 5.1  | Флаги для таблиц Заявки, Заявки на внебиржевые сделки, Сделки, Сделки для исполнения..... | 78 |
| 5.2  | Флаги для таблицы Обезличенные сделки.....  | 79 |
| 5.3  | Флаги для таблицы Стоп-заявки .....   | 79 |
| 5.4  | Дополнительные флаги для таблицы Стоп-заявки .....  | 80 |

|     |   |     |
|-----|---|-----|
| 5.5 | Обязательства и требования по активам .....   | 80  |
| 6.  | Функции для работы с битовыми масками в структурах данных .....                         | 81  |
| 6.1 | bit.tohex .....   | 81  |
| 6.2 | bit.bnot .....  | 81  |
| 6.3 | bit.band .....  | 81  |
| 6.4 | bit.bor .....   | 81  |
| 6.5 | bit.bxor .....  | 82  |
| 7.  | Индикаторы технического анализа .....   | 82  |
| 7.1 | Общие сведения .....  | 82  |
| 7.2 | Функции и глобальные переменные скрипта индикатора .....                                | 83  |
| 8.  | Потокобезопасные функции для работы с таблицами Lua .....                               | 92  |
| 9.  | Приложения .....  | 93  |
|     | Приложение 1. Пример скрипта на языке Lua .....   | 93  |
|     | Приложение 2. Примеры сортировки в таблицах .....                                       | 100 |
|     | Приложение 3. Примеры обработки событий для таблиц .....                                | 101 |
|     | Приложение 4. Примеры использования параметра «params» в функции<br>«SearchItems» ..... | 111 |

Ваши пожелания и комментарии к данной Инструкции

направляйте по электронной почте на адрес: [quiksupport@arqatech.com](mailto:quiksupport@arqatech.com)

# 1. О программе

Интерпретатор языка Lua (**QLua**) – это библиотека, которая предоставляет пользователю возможность взаимодействия с Рабочим местом QUIK при помощи скриптов, созданных на языке Lua. Подробнее о Lua на сайте [www.lua.org](http://www.lua.org).

## 1.1 Возможности

**QLua** обеспечивает доступ к внутренним данным Рабочего места QUIK и отправку клиентских транзакций из скрипта Lua на сервер QUIK. **QLua** обеспечивает (в отличие от языка QPILE) асинхронную обработку данных в скрипте по мере получения, а так же возможность непрерывно опрашивать клиентский терминал для получения новых данных. Для асинхронной обработки используются специальные функции обратного вызова (далее callback), которые описываются в скрипте.

**QLua** обеспечивает обработку следующих событий от Рабочего места QUIK:

- получение новых данных;
- подключение к серверу QUIK;
- отключение от сервера QUIK;
- остановка выполнения скрипта;
- закрытие Рабочего места QUIK.

Все перечисленные события могут вызывать соответствующий callback в скрипте.

Дополнительной возможностью является загрузка в скрипт библиотек, написанных на других языках программирования.

**Некорректная работа сторонних библиотек, загруженных скриптом, может приводить к ошибкам в работе терминала QUIK.**

## 1.2 Подготовка к работе

**QLua** является дополнительным компонентом Рабочего места QUIK.

Работу **QLua** обеспечивает файл qlua.dll, который должен находиться в одной папке с файлами Рабочего места QUIK, например, C:\Program Files\QUIK.

## 1.3 Работа с программой

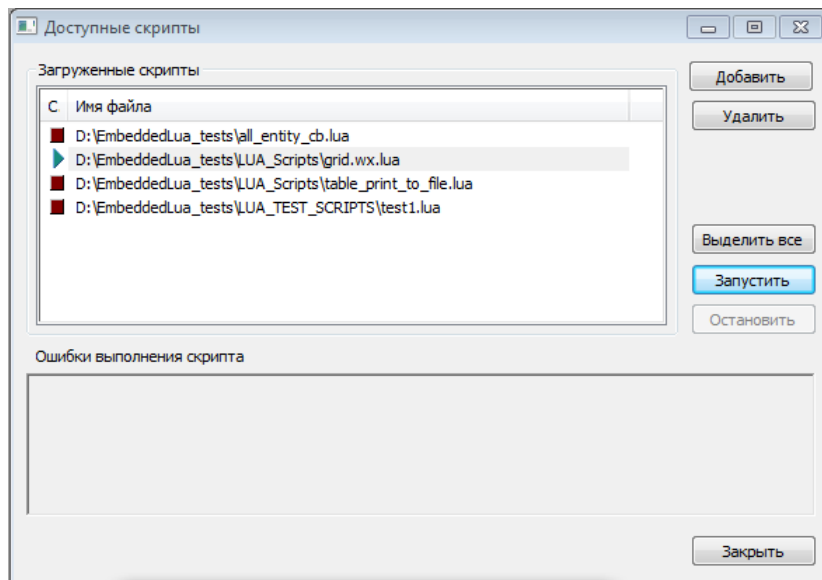
Доступ к функциям **QLua** осуществляется из главного меню программы. После установки компонента, в меню программы появляется дополнительный пункт **Сервисы / Lua скрипты...**

### 1.3.1 Назначение

Форма «Доступные скрипты» предназначена для добавления, удаления, перезагрузки и запуска скриптов.

### 1.3.2 Настройки окна

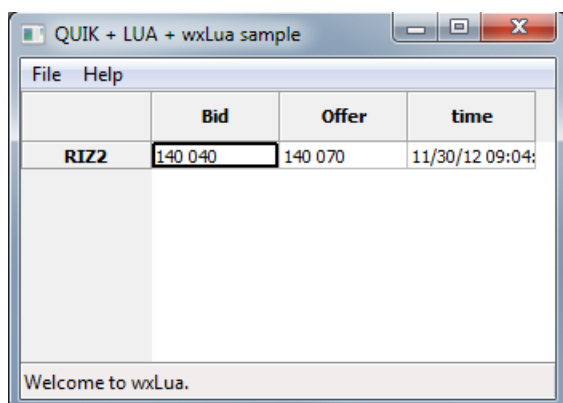
Настройки в этом окне:



- «Добавить» – загрузить новый скрипт из файла,
- «Удалить» – удалить загруженный скрипт,
- «Выделить все» – выделить все доступные скрипты,
- «Запустить» – запустить выполнение загруженных скриптов,
- «Остановить» – остановить выполнение выбранного скрипта.

Активность кнопок «Запустить» и «Остановить» зависит от текущего состояния скрипта.

Пример результата выполнения скрипта:



Ошибки, возникающие при выполнении скрипта, отображаются в соответствующей области окна.

## 2. Функции терминала, доступные в скрипте Lua

### 2.1 Сервисные функции

#### 2.1.1 isConnected

Функция предназначена для определения состояния подключения клиентского места к серверу. Возвращает «1», если клиентское место подключено и «0», если не подключено.

Формат вызова:

NUMBER isConnected()

#### 2.1.2 getScriptPath

Функция возвращает путь, по которому находится запускаемый скрипт, без завершающего обратного следа («\»). Например, C:\QuikFront\Scripts.

Формат вызова:

STRING getScriptPath()

Пример:

```
path = getScriptPath()
```

#### 2.1.3 getInfoParam

Функция возвращает значения параметров информационного окна (пункт меню **Система / О программе / Информационное окно...**).

Формат вызова:

STRING getInfoParam (STRING param\_name)

Параметр «param\_name» может принимать значения, представленные в таблице.

| Значение параметра | Описание         |
|--------------------|------------------|
| VERSION            | Версия программы |
| TRADEDATE          | Дата торгов      |
| SERVERTIME         | Время сервера    |

| Значение параметра | Описание               |
|--------------------|------------------------|
| LASTRECORDTIME     | Время последней записи |
| NUMRECORDS         | Число записей          |
| LASTRECORD         | Последняя запись       |

| Значение параметра | Описание               |
|--------------------|------------------------|
| LATERECORD         | Отставшая запись       |
| CONNECTION         | Соединение             |
| IPADDRESS          | IP-адрес сервера       |
| IPPORT             | Порт сервера           |
| IPCOMMENT          | Описание соединения    |
| SERVER             | Описание сервера       |
| SESSIONID          | Идентификатор сессии   |
| USER               | Пользователь           |
| USERID             | ID пользователя        |
| ORG                | Организация            |
| MEMORY             | Занято памяти          |
| LOCALTIME          | Текущее время          |
| CONNECTIONTIME     | Время на связи         |
| MESSAGESENT        | Передано сообщений     |
| ALLSENT            | Передано всего байт    |
| BYTESENT           | Передано полезных байт |

| Значение параметра | Описание                              |
|--------------------|---------------------------------------|
| BYTESPERSECSSENT   | Передано за секунду                   |
| MESSAGESRECV       | Принято сообщений                     |
| BYTESRECV          | Принято полезных байт                 |
| ALLRECV            | Принято всего байт                    |
| BYTESPERSECRECV    | Принято за секунду                    |
| AVGSENT            | Средняя скорость передачи             |
| AVGRECV            | Средняя скорость приема               |
| LASTPINGTIME       | Время последней проверки связи        |
| LASTPINGDURATION   | Задержка данных при обмене с сервером |
| AVGPINGDURATION    | Средняя задержка данных               |
| MAXPINGTIME        | Время максимальной задержки           |
| MAXPINGDURATION    | Максимальная задержка данных          |

Пример:

```
function main( )
    params = {"VERSION", "TRADEDATE", "SERVERTIME",
              "LASTRECORDTIME", "NUMRECORDS", "LASTRECORD", "LATERECORD",
              "CONNECTION", "IPADDRESS", "IPPORT", "IPCOMMENT",
              "SERVER", "SESSIONID", "USER", "USERID", "ORG", "MEMORY",
              "LOCALTIME", "CONNECTIONTIME", "MESSAGESENT", "ALLSENT",
              "BYTESENT", "BYTESPERSECSSENT", "MESSAGESRECV", "BYTESRECV",
              "ALLRECV", "BYTESPERSECRECV", "AVGSENT", "AVGRECV",
              "LASTPINGTIME", "LASTPINGDURATION", "AVGPINGDURATION",
              "MAXPINGTIME", "MAXPINGDURATION"}

    file = io.open("res.txt", "w+t")
    for key,v in ipairs(params) do
```



```

        file:write(v .. " = " .. GetInfoParam(v) .. "\n")
    end
    file:close()
end

```

#### 2.1.4 message




Функция отображает сообщения в терминале QUIK. Возвращает «nil» при ошибке выполнения или при обнаружении ошибки во входных параметрах. В остальных случаях возвращает «1».

**Максимальная длина сообщений об ошибках выполнения скрипта и строк, передаваемых в функцию message(), составляет 900 символов.**

Формат вызова:

NUMBER message(String message, NUMBER icon\_type)

Параметры:

| Параметр  | Тип    | Описание  |
|-----------|--------|---|
| message   | STRING | Строка, отображаемая в окне сообщений терминала QUIK  |
| icon_type | NUMBER | Тип отображаемой иконки в сообщении. Возможные значения и вид иконки: <ul style="list-style-type: none"> <li>— «1» (по умолчанию) – ;</li> <li>— «2» – ;</li> <li>— «3» – ;</li> </ul> Необязательный параметр |

Пример:

```

message("test message")
message("test message", 1)
message("test\nmessage", 2)
message("connection state is " .. tostring(isConnected()), 3)

```

#### 2.1.5 sleep

Функция приостанавливает выполнение скрипта. Возвращает «nil» при обнаружении ошибки во входных параметрах. В случае успешного завершения возвращает время ожидания, заданное параметром «time».

Формат вызова:

NUMBER sleep(NUMBER time)

Параметры:

| Параметр | Тип    | Описание   |
|----------|--------|--|
| time     | NUMBER | Время, на которое приостанавливается выполнение, в миллисекундах |

Пример:

```
Sleep(1000)    приостановка выполнения скрипта на одну секунду
```

**Функцию sleep не рекомендуется использовать в функциях обратного вызова.**

### 2.1.6 getWorkingFolder

Функция возвращает путь, по которому находится файл **info.exe**, исполняющий данный скрипт, без завершающего обратного следа («\»). Например, c:\QuikFront.

Формат вызова:

STRING getWorkingFolder()

Пример:

```
path = getWorkingFolder
```

### 2.1.7 PrintDbgStr

Функция для вывода отладочной информации.

Формат вызова:

PrintDbgStr(STRING s)

Пример:

```
function main()
    PrintDbgStr("test1")
    PrintDbgStr("test2")
    PrintDbgStr("dbg from " .. getScriptPath())
```

## 2.2 Функции обратного вызова

Функции, представленные в этом разделе, вызываются при получении терминалом QUIK данных от сервера и при обработке других внешних событий.

**Функции обратного вызова обрабатываются в основном потоке терминала QUIK. Поэтому пользователю необходимо оптимизировать время исполнения таких функций.**

### 2.2.1 OnFirm

Функция вызывается терминалом QUIK при получении описания новой фирмы от сервера.

Формат вызова:

```
OnFirm(TABLE firm)
```

Параметры:

| Параметр | Тип   | Описание                                    |
|----------|-------|---|
| firm     | TABLE | <a href="#">Таблица с параметрами фирмы</a> |

### 2.2.2 OnAllTrade

Функция вызывается терминалом QUIK при получении обезличенной сделки.

Формат вызова:

```
OnAllTrade(TABLE alltrade)
```

Параметры:

| Параметр | Тип   | Описание  |
|----------|-------|---|
| alltrade | TABLE | <a href="#">Таблица с параметрами обезличенной сделки</a> |

### 2.2.3 OnTrade

Функция вызывается терминалом QUIK при получении сделки.

Формат вызова:

OnTrade(TABLE trade)

Параметры:

| Параметр | Тип   | Описание                                     |
|----------|-------|--|
| trade    | TABLE | <a href="#">Таблица с параметрами сделки</a> |

### 2.2.4 OnOrder

Функция вызывается терминалом QUIK при получении новой заявки или при изменении параметров существующей заявки.

Формат вызова:

OnOrder(TABLE order)

Параметры:

| Параметр | Тип   | Описание                                     |
|----------|-------|--|
| order    | TABLE | <a href="#">Таблица с параметрами заявки</a> |

### 2.2.5 OnAccountBalance

Функция вызывается терминалом QUIK при получении изменений текущей позиции по счету.

Формат вызова:

OnAccountBalance(TABLE acc\_bal)

Параметры:

| Параметр | Тип   | Описание   |
|----------|-------|--|
| acc_bal  | TABLE | <a href="#">Таблица с текущими позициями по счетам</a> |

### 2.2.6 OnFuturesLimitChange

Функция вызывается терминалом QUIK при получении изменений ограничений по срочному рынку.



Формат вызова:

```
OnFuturesLimitChange(TABLE fut_limit)
```

Параметры:

| Параметр  | Тип   | Описание   |
|-----------|-------|--|
| fut_limit | TABLE | <a href="#">Таблица с текущими значениями лимита по срочному рынку</a> |

### 2.2.7 OnFuturesLimitDelete

Функция вызывается терминалом QUIK при удалении лимита по срочному рынку.

Формат вызова:

```
OnFuturesLimitDelete(TABLE lim_del)
```

Параметры:

| Параметр | Тип   | Описание  |
|----------|-------|---|
| lim_del  | TABLE | <a href="#">Таблица с параметрами удаляемого лимита по срочному рынку</a> |

### 2.2.8 OnFuturesClientHolding

Функция вызывается терминалом QUIK при изменении позиции по срочному рынку.

Формат вызова:

```
OnFuturesClientHolding(TABLE fut_pos)
```

Параметры:

| Параметр | Тип   | Описание  |
|----------|-------|---|
| fut_pos  | TABLE | <a href="#">Таблица с описанием позиции по срочному рынку</a> |

### 2.2.9 OnMoneyLimit

Функция вызывается терминалом QUIK при получении изменений по денежному лимиту клиента.

Формат вызова:

```
OnMoneyLimit(TABLE mlimit)
```

Параметры:

| Параметр | Тип   | Описание   |
|----------|-------|--|
| mlimit   | TABLE | <a href="#">Таблица с текущими значениями денежного лимита</a> |

### 2.2.10 OnMoneyLimitDelete

Функция вызывается терминалом QUIK при удалении денежного лимита.

Формат вызова:

```
OnMoneyLimitDelete(TABLE mlimit_del)
```

Параметры:

| Параметр   | Тип   | Описание  |
|------------|-------|---|
| mlimit_del | TABLE | <a href="#">Таблица с параметрами удаляемого денежного лимита</a> |

### 2.2.11 OnDepoLimit

Функция вызывается терминалом QUIK при получении изменений лимита по бумагам.

Формат вызова:

```
OnDepoLimit(TABLE dlimit)
```

Параметры:

| Параметр | Тип   | Описание  |
|----------|-------|---|
| dlimit   | TABLE | <a href="#">Таблица с текущими значениями лимита по бумагам</a> |

### 2.2.12 OnDepoLimitDelete

Функция вызывается терминалом QUIK при удалении клиентского лимита по бумагам.

Формат вызова:

```
OnDepoLimitDelete(TABLE dlimit_del)
```

Параметры:

| Параметр   | Тип   | Описание   |
|------------|-------|--|
| dlimit_del | TABLE | <a href="#">Таблица с параметрами удаляемого лимита по бумагам</a> |

### 2.2.13 OnAccountPosition

Функция вызывается терминалом QUIK при изменении денежной позиции по счету.

Формат вызова:

```
OnAccountPosition(TABLE acc_pos)
```

Параметры:

| Параметр | Тип   | Описание  |
|----------|-------|---|
| acc_pos  | TABLE | <a href="#">Таблица с текущими значениями денежной позиции по счету</a> |

### 2.2.14 OnNegDeal

Функция вызывается терминалом QUIK при получении заявки на внебиржевые сделки.

Формат вызова:

```
OnNegDeal(TABLE neg_deals)
```

Параметры:

| Параметр  | Тип   | Описание   |
|-----------|-------|--|
| neg_deals | TABLE | <a href="#">Таблица с параметрами заявки на внебиржевые сделки</a> |

### 2.2.15 OnNegTrade

Функция вызывается терминалом QUIK при получении сделки для исполнения.

Формат вызова:

```
OnNegTrade(TABLE neg_trade)
```

Параметры:

| Параметр  | Тип   | Описание  |
|-----------|-------|---|
| neg_trade | TABLE | <a href="#">Таблица с параметрами сделки для исполнения</a> |

### 2.2.16 OnStopOrder

Функция вызывается терминалом QUIK при получении новой стоп-заявки или при изменении параметров существующей стоп-заявки.

Формат вызова:

OnStopOrder(TABLE stop\_order)

Параметры:

| Параметр   | Тип   | Описание  |
|------------|-------|---|
| stop_order | TABLE | <a href="#">Таблица с параметрами стоп-заявки</a> |

### 2.2.17 OnTransReply

Функция вызывается терминалом QUIK при получении ответа на транзакцию пользователя.

Формат вызова:

OnTransReply(TABLE trans\_reply)

Параметры:

| Параметр    | Тип   | Описание                                       |
|-------------|-------|--|
| trans_reply | TABLE | <a href="#">Таблица с описанием транзакций</a> |

### 2.2.18 OnParam

Функция вызывается терминалом QUIK при изменении текущих параметров.

OnParam (STRING class\_code, STRING sec\_code)

Параметры:

| Параметр   | Тип    | Описание   |
|------------|--------|------------|
| class_code | STRING | Код класса |
| sec_code   | STRING | Код бумаги |

**При вызове данной функции пользователь может вызвать функцию `getParamEx` и получить значение нужного параметра.**

Пример:

```
function OnParam( class, sec )  
if class == "SPBFUT" and sec == "RIZ2" then
```



```
tbid = getParamEx(class, sec, "bid")
if tbid.param_value >=130000 then
\
\
End
```

### 2.2.19 OnQuote

Функция вызывается терминалом QUIK при получении изменения стакана котировок.

OnQuote(String class\_code, String sec\_code)

**После вызова данной функции запросить нужную таблицу котировок можно с помощью функции getQuoteLevel2().**

Пример:

```
function OnQuote(class, sec )

if class == "SPBFUT" and sec == "RIZ2" then
ql2 = getQuoteLevel2(class, sec)
end
```

### 2.2.20 OnDisconnected

Функция вызывается терминалом QUIK при отключении от сервера QUIK.

OnDisconnected()

### 2.2.21 OnConnected

Функция вызывается терминалом QUIK при установлении связи с сервером QUIK и получении терминалом описания хотя бы одного класса. Если в течение торгового дня терминал получает новый класс, то функция вызывается еще раз, при этом параметр вызова flag принимает значение «false».

OnConnected(BOOLEAN flag)

### 2.2.22 OnCleanUp

Функция вызывается терминалом QUIK при смене сессии.

OnCleanUp()

**Под сменой сессии подразумевается изменение идентификатора сессии при подключении к серверу QUIK.**

### 2.2.23 OnClose

Функция вызывается перед закрытием терминала QUIK и при выгрузке файла qlua.dll.

OnClose()

**Под выгрузкой файла qlua.dll подразумевается отключение плагина QLua в окне «Версии компонентов и плагинов» (см. п. 1.9. Раздела 1 Руководства пользователя QUIK).**

### 2.2.24 OnStop

Функция вызывается терминалом QUIK при остановке скрипта из диалога управления и при закрытии терминала QUIK.

[NUMBER time\_out] OnStop(NUMBER signal)

Функция возвращает количество миллисекунд, которое дается скрипту на завершение работы. Если функция не возвращает число, то таймаут завершения работы скрипта остается равным 5 секундам.

**По истечении интервала времени, данного скрипту на завершение работы, функция main() завершается принудительно. При этом возможна потеря системных ресурсов.**

Пример:

```
function OnStop(s)
    stopped = true
    return 3000 -- задается таймаут в 3 секунды
end
function OnStop(s)
    stopped = true
    return '3000' -- возвращаемое значение не число, таймаут остается равным 5 секундам
end
```

### 2.2.25 OnInit

Функция вызывается терминалом QUIK перед вызовом функции **main()**. В качестве параметра принимает значение полного пути к запускаемому скрипту.

OnInit(STRING script\_path)

В данной функции пользователь имеет возможность инициализировать все необходимые переменные и библиотеки перед запуском основного потока `main()`.

### 2.2.26 `main`

Функция, реализующая основной поток выполнения в скрипте. Для ее выполнения терминал QUIK создает отдельный поток. Скрипт считается работающим, пока работает функция `main()`. При завершении работы функции `main()` скрипт переходит в состояние «остановлен». Если скрипт находится в состоянии «остановлен», то не происходит вызовов функций обработки событий терминала QUIK, содержащихся в этом скрипте.

Уведомления передаются в скрипт только в том случае, если параметры по данной бумаге заказаны с сервера (в рамках умного заказа или вручную через диалог Система/Заказ данных) или если открыт соответствующий стакан.

Заказ данных с сервера также можно выполнить с помощью функций `QLua` (заказ стакана котировок – см. п. [3.17](#), заказ параметров Таблицы текущих торгов – см. п. [3.18](#))

## 3. Функции взаимодействия скрипта Lua и Рабочего места QUIK

### 3.1 Функции для обращения к строкам произвольных таблиц QUIK

Функции из этой группы предназначены для доступа к данным, содержащимся в таблицах Рабочего места QUIK.

#### 3.1.1 `getItem`

Функция возвращает таблицу Lua, содержащую информацию о данных из строки с номером «Index» из таблицы с именем «TableName».

`getItem (STRING TableName, DOUBLE Index)`

Индекс элементов в таблице начинается с «0».

#### 3.1.2 `getOrderByNumber`

Функция возвращает таблицу Lua, содержащую описание параметров [Таблицы заявок](#) и индекс заявки в хранилище терминала.

`getOrderByNumber (STRING class_code, NUMBER order_id)`

Если заявка с указанным номером не существует, то возвращаемые параметры – «nil».

### 3.1.3 getNumberOf

Функция возвращает количество записей в таблице «TableName».

```
getNumberOf(String TableName)
```

### 3.1.4 SearchItems

Функция позволяет реализовать быструю выборку элементов из хранилища терминала и возвращает таблицу с индексами элементов, удовлетворяющих условию поиска.

```
TABLE SearchItems(String table_name, NUMBER start_index, NUMBER end_index,  
FUNCTION fn [, STRING params])
```

Параметры:

- **table\_name** – строка, определяющая таблицу для поиска;
- **start\_index** – индекс стартового элемента для поиска;
- **end\_index** – индекс конечного элемента поиска;
- **fn** – функция обратного вызова, возвращающая true или false. Входные параметры функции определяются наличием и значением параметра params;
- **params** – определяет список полей элемента таблицы **table\_name** для передачи в функцию **fn**. Поля задаются через запятую, пробелы игнорируются. Если строка **params** не задана, то в функцию **fn** элемент из хранилища терминала передается полностью. Необязательный параметр.

Примеры с использованием **params** приведены в [Приложении 4](#).

### 3.1.5 Таблицы, используемые в функциях «getItem», «getNumberOf», «getOrderByNumber» и «SearchItems»

| TableName         | Таблица                             |
|-------------------|-------------------------------------|
| firms             | <a href="#">Фирмы</a>               |
| classes           | <a href="#">Классы</a>              |
| securities        | <a href="#">Инструменты</a>         |
| trade_accounts    | <a href="#">Торговые счета</a>      |
| client_codes      | * Коды клиентов                     |
| all_trades        | <a href="#">Обезличенные сделки</a> |
| account_positions | <a href="#">Денежные позиции</a>    |
| orders            | <a href="#">Заявки</a>              |

| TableName              | Таблица   |
|------------------------|---|
| futures_client_holding | <a href="#">Позиции по клиентским счетам (фьючерсы)</a> |
| futures_client_limits  | <a href="#">Лимиты по фьючерсам</a>                     |
| money_limits           | <a href="#">Лимиты по денежным средствам</a>            |
| depo_limits            | <a href="#">Лимиты по бумагам</a>                       |
| trades                 | <a href="#">Сделки</a>                                  |
| stop_orders            | <a href="#">Стоп-заявки</a>                             |
| neg_deals              | <a href="#">Заявки на внебиржевые сделки</a>            |
| neg_trades             | <a href="#">Сделки для исполнения</a>                   |
| neg_deal_reports       | <a href="#">Отчеты по сделкам для исполнения</a>        |
| firm_holding           | <a href="#">Текущие позиции по бумагам</a>              |
| account_balance        | <a href="#">Позиции по клиентским счетам</a>            |
| ccp_holdings           | <a href="#">Обязательства и требования по активам</a>   |

\* – функция `getNumberOf(«client_codes»)` возвращает количество доступных кодов клиента в терминале, а функция `getItem(«client_codes», i)` – строку, содержащую клиентский код с индексом `i`, где `i` может принимать значения от 0 до `getNumberOf(«client_codes»)-1`

Пример:

```
function main()
  n = getNumberOf("orders")
  order={}
  message("total ".. tostring(n) .. " of all orders", 1)
  for i=0,n-1 do
    order = getItem("orders", i)
    message("order: num=" .. tostring(order["order_num"]) .. " qty=" ..
      tostring(order["qty"]) .. " value=" .. tostring(order["value"]), 1)
  end
end
```

В результате выполнения данного скрипта выводится информация обо всех заявках.

## 3.2 Функции для обращения к спискам доступных параметров

### 3.2.1 getClassesList

Функция предназначена для получения списка кодов классов, переданных с сервера в ходе сеанса связи. Коды классов в списке разделяются запятой «,». В конце полученной строки всегда дописывается символ «,».

```
getClassesList ()
```

Пример:

```
list = getClassesList ()
```

В результате выполнения приведенной строки кода, переменная list содержит строку вида:

```
FUTEVN, OPTEVN, RTSSTEVN, OPTEXP, USDRUB, REPORTSST,
```

### 3.2.2 getClassInfo

Функция предназначена для получения информации о классе.

```
getClassInfo (STRING)
```

Возвращает таблицу Lua, содержащую описание класса:

| Параметр | Тип    | Описание                       |
|----------|--------|--------------------------------|
| firmid   | NUMBER | Код фирмы                      |
| name     | STRING | Наименование класса            |
| code     | STRING | Код класса                     |
| nparams  | NUMBER | Количество параметров в классе |
| nsecs    | NUMBER | Количество бумаг в классе      |

### 3.2.3 getClassSecurities

Функция предназначена для получения списка кодов бумаг для списка классов, заданного списком кодов. Коды бумаг в списке разделяются запятой «,». В конце полученной строки всегда дописывается символ «,».

getClassSecurities (STRING)

Пример:

```
sec_list = getClassSecurities""SPBFU""")
```

В результате выполнения приведенной строки кода, переменная `sec_list` содержит строку вида:

```
RSH3,VBZ2,O4Z2,O2Z2,SiM3,SiH3,SiF3,RIH3,RIM3,LKH3,LKZ2,GDZ2,GMZ2,GZH3,GZZ2,EuZ2,EDZ2,  
SiZ2,RIZ2,
```

## 3.3 Функция для получения информации по денежным средствам

### 3.3.1 `getMoney`

Функция предназначена для получения информации по денежным лимитам.

`getMoney (STRING client_code, STRING firmid, STRING tag, STRING currcode)`

Функция возвращает таблицу Lua с параметрами:

| Параметр  | Тип    | Описание   |
|---|--------|--|
| <code>money_open_limit</code>                     | NUMBER | Входящий лимит по денежным средствам                             |
| <code>money_limit_locked_nonmarginal_value</code> | NUMBER | Стоимость немаржинальных бумаг в заявках на покупку              |
| <code>money_limit_locked</code>                   | NUMBER | Заблокированное в заявках на покупку количество денежных средств |
| <code>money_open_balance</code>                   | NUMBER | Входящий остаток по денежным средствам                           |
| <code>money_current_limit</code>                  | NUMBER | Текущий лимит по денежным средствам                              |
| <code>money_current_balance</code>                | NUMBER | Текущий остаток по денежным средствам                            |
| <code>money_limit_available</code>                | NUMBER | Доступное количество денежных средств                            |

### 3.3.2 `getMoneyEx`

Функция предназначена для получения информации по денежным лимитам указанного типа.

`TABLE getMoneyEx(STRING firm_id, STRING client_code, STRNIG tag, STRING curr_code, NUMBER limit_kind)`

Функция возвращает таблицу Lua с параметрами [Таблицы лимитов по денежным средствам](#).

В случае ошибки функция возвращает «nil».

## 3.4 Функция для получения информации по бумажным лимитам

### 3.4.1 getDepo

Функция предназначена для получения информации по бумажным лимитам.

getDepo (STRING client\_code, STRING firmid, STRING sec\_code, STRING account)

Функция возвращает таблицу Lua с параметрами:

| Параметр                    | Тип    | Описание  |
|-----------------------------|--------|---|
| depo_limit_locked_buy_value | NUMBER | Стоимость ценных бумаг, заблокированных на покупку        |
| depo_current_balance        | NUMBER | Текущий остаток по бумагам                                |
| depo_limit_locked_buy       | NUMBER | Количество лотов ценных бумаг, заблокированных на покупку |
| depo_limit_locked           | NUMBER | Заблокированное Количество лотов ценных бумаг             |
| depo_limit_available        | NUMBER | Доступное количество ценных бумаг                         |
| depo_current_limit          | NUMBER | Текущий лимит по бумагам                                  |
| depo_open_balance           | NUMBER | Входящий остаток по бумагам                               |
| depo_open_limit             | NUMBER | Входящий лимит по бумагам                                 |

### 3.4.2 getDepoEx

Функция предназначена для получения информации по бумажным лимитам указанного типа.

TABLE getDepoEx(STRING firm\_id, STRING client\_code, STRING sec\_code,  
STRING acc\_id, NUMBER limit\_kind)

Функция возвращает таблицу Lua с параметрами [Таблицы лимитов по бумагам](#).

В случае ошибки функция возвращает «nil».



## 3.5 Функция для получения информации по фьючерсным лимитам

### 3.5.1 getFuturesLimit

Функция предназначена для получения информации по фьючерсным лимитам.

```
TABLE getFuturesLimit(String firm_id, String acc_id, NUMBER limit_type, String curr_code)
```

Если необходимо получить информацию по фьючерсному лимиту без валюты, то в качестве `curr_code` задается пустая строка.

Функция возвращает таблицу Lua с параметрами [фьючерсного лимита](#).

В случае ошибки функция возвращает «nil».

## 3.6 Функция для получения информации по фьючерсным позициям

### 3.6.1 getFuturesHolding

Функция предназначена для получения информации по фьючерсным позициям.

```
TABLE getFuturesHolding(String firm_id, String acc_id, String sec_code, NUMBER pos_type)
```

Функция возвращает таблицу Lua с параметрами [Таблицы позиций по фьючерсам](#).

В случае ошибки функция возвращает «nil».

## 3.7 Функция для получения информации по инструменту

### 3.7.1 getSecurityInfo

Функция предназначена для получения информации по бумаге.

```
getSecurityInfo (String class_code, String sec_code)
```

Функция возвращает таблицу Lua с параметрами [Таблицы инструментов](#).

## 3.8 Функция для получения даты торговой сессии

### 3.8.1 getTradeDate

Возвращает дату текущей торговой сессии.

```
getTradeDate ()
```

Функция возвращает таблицу Lua с параметрами:

| Параметр | Тип    | Описание                                 |
|----------|--------|--|
| date     | STRING | Торговая дата в виде строки <ДД.ММ.ГГГГ> |
| year     | NUMBER | Год                                      |
| month    | NUMBER | Месяц                                    |
| day      | NUMBER | День                                     |

## 3.9 Функция для получения стакана по указанному классу и бумаге

### 3.9.1 getQuoteLevel2

Функция предназначена для получения стакана по указанному классу и бумаге.

```
getQuoteLevel2 (STRING class_code, STRING sec_code)
```

Функция возвращает таблицу Lua с параметрами:

| Параметр    | Тип    | Описание                        |
|-------------|--------|---------------------------------|
| bid_count   | STRING | Количество котировок покупки    |
| offer_count | STRING | Количество котировок продажи    |
| bid         | TABLE  | Котировки спроса (покупки)      |
| offer       | TABLE  | Котировки предложений (продажи) |

Таблицы «bid» и «offer» имеют следующую структуру:

| Параметр | Тип    | Описание               |
|----------|--------|------------------------|
| price    | STRING | Цена покупки / продажи |
| quantity | STRING | Количество в лотах     |

## 3.10 Функции для работы с графиками

### 3.10.1 `getLinesCount`

Функция предназначена для получения количества линий в графике (индикаторе) по выбранному идентификатору:

NUMBER `getLinesCount` (STRING `tag`)

Возвращает число – количество линий на графике.

### 3.10.2 `getNumCandles`

Функция предназначена для получения информации о количестве свечек по выбранному идентификатору:

NUMBER `getNumCandles` (STRING `tag`)

Возвращает число – количество свечек по выбранному идентификатору.

### 3.10.3 `getCandlesByIndex`

Функция предназначена для получения информации о свечках по идентификатору (заказ данных для построения графика плагин не осуществляет, поэтому для успешного доступа нужный график должен быть открыт):

Формат вызова:

TABLE `t`, NUMBER `n`, STRING `l` `getCandlesByIndex` (STRING `tag`, NUMBER `line`, NUMBER `first_candle`, NUMBER `count`)

Параметры:

- **tag** – строковый идентификатор графика или индикатора,
- **line** – номер линии графика или индикатора. Первая линия имеет номер 0,
- **first\_candle** – индекс первой свечки. Первая (самая левая) свечка имеет индекс 0,
- **count** – количество запрашиваемых свечек.

Возвращаемые значения:

- **t** – таблица, содержащая запрашиваемые свечки,
- **n** – количество свечек в таблице **t**,
- **l** – легенда (подпись) графика.

### 3.10.4 `CreateDataSource`

Функция предназначена для создания таблицы Lua и позволяет работать со свечками, полученными с сервера QUIK, а также реагировать на их изменение.

Формат вызова:

```
TABLE data_source, STRING error_desc CreateDataSource (STRING class_code, STRING sec_code, NUMBER interval, [, STRING param])
```

Параметры:

- **class\_code** – код класса,
- **sec\_code** – код бумаги,
- **interval** – интервал запрашиваемого графика,
- **param** – необязательный параметр. Если параметр не задан, то заказываются данные на основании Таблицы обезличенных сделок, если задан – данные по этому параметру.

Функция возвращает таблицу **data\_source** в случае успешного завершения. Если указан неверный код класса или параметр, то возвращается «nil». При этом **error\_desc** содержит описание ошибки.

Список констант для передачи в параметр **interval**:

| Параметр      | Значение интервала |
|---------------|--------------------|
| INTERVAL_TICK | Тиковые данные     |
| INTERVAL_M1   | 1 минута           |
| INTERVAL_M2   | 2 минуты           |
| INTERVAL_M3   | 3 минуты           |
| INTERVAL_M4   | 4 минуты           |
| INTERVAL_M5   | 5 минут            |
| INTERVAL_M6   | 6 минут            |
| INTERVAL_M10  | 10 минут           |
| INTERVAL_M15  | 15 минут           |
| INTERVAL_M20  | 20 минут           |
| INTERVAL_M30  | 30 минут           |
| INTERVAL_H1   | 1 час              |
| INTERVAL_H2   | 2 часа             |
| INTERVAL_H4   | 4 часа             |
| INTERVAL_D1   | 1 день             |
| INTERVAL_W1   | 1 неделя           |

| Параметр     | Значение интервала |
|--------------|--------------------|
| INTERVAL_MN1 | 1 месяц            |

Функция **CreateDataSource** возвращает таблицу Lua с параметрами:

| Параметр          | Тип      | Описание   |
|-------------------|----------|--|
| SetUpdateCallback | function | Позволяет задать пользователю функцию обратного вызова для обработки изменившихся свечек |
| O                 | function | Получить значение Open для указанной свечи   |
| H                 | function | Получить значение High для указанной свечи   |
| L                 | function | Получить значение Low для указанной свечи  |
| C                 | function | Получить значение Close для указанной свечи  |
| V                 | function | Получить значение Volume для указанной свечи   |
| T                 | function | Получить значение Time для указанной свечи   |
| Size              | function | Возвращает текущий размер (количество свечек в источнике данных)                         |
| Close             | function | Удаляет источник данных, отписывается от получения данных                                |

Пример:

```
ds1 = CreateDataSource("SPBFUT", "RIU3", INTERVAL_M1, "last")
ds2 = CreateDataSource("SPBFUT", "RIU3", INTERVAL_M1)
ds3 = CreateDataSource("SPBFUT", "RIU3", INTERVAL_M1, "bid")
```

Подробное описание каждой функции приведено ниже.

### Функция SetUpdateCallback

Формат вызова:

BOOLEAN res SetUpdateCallback (FUNCTION callback\_function)

В качестве параметра принимает функцию обратного вызова.

Формат функции обратного вызова:

function call\_back(NUMBER index)

Параметры:

- **index** – номер изменившейся свечки. Индексы свечек начинаются с 1.

Функция возвращает «true» в случае успешного завершения, иначе – «false».

Пример получения времени из свечки:

```
function cb( index )
local t = ds:T(index)
local _str = string.format("#%d of %d\t%.4f\t%.4f\t%.4f\t%.4f %02d.%02d.%04d
%02d:%02d:%02d.%04d",
index, ds:Size(), ds:O(index), ds:H(index), ds:L(index),
ds:C(index), ds:V(index),
t.day, t.month, t.year, t.hour, t.min, t.sec, t.ms)
Log(file, _str)
end
ds: SetUpdateCallback (cb)
```

### Функции O, H, L, C, V, T

Функции в качестве параметра принимают индекс свечи и возвращают соответствующее значение. Время свечи возвращается с точностью до миллисекунд в виде таблицы с полями:

{year, month, day, week\_day, hour, min, sec, ms, count}

Где:

- **count** – количество тиковых интервалов в секунду. Может принимать значения от «1» до «10000» включительно.

Пример:

```
Open = ds:O(1)
High = ds:H(1)
Low = ds:L(1)
Close = ds:C(1)
Volume = ds:V(1)
week_day = ds:T(1).week_day
count = ds:T(1).count
```

### Функция Size

Функция возвращает текущее количество свечек в источнике данных.

Формат вызова:

NUMBER Size()

Пример:

```
ds:Size()
```

### Функция Close

Функция закрывает источник данных, и терминал прекращает получать данные с сервера.

Формат вызова:

BOOLEAN Close()

Пример:

```
ds:Close()
```

Функция возвращает «true» в случае успешного завершения.

### Функция SetEmptyCallback

Функция позволяет получать данные с сервера.

Формат вызова:

BOOLEAN SetEmptyCallback()

Функция возвращает «true» в случае успешного завершения, иначе – «false».

Пример:

```
ds:SetEmptyCallback()
```

## 3.11 Функции для работы с заявками

Функции предназначены для создания заявок и отправки их в торговую систему.

### 3.11.1 sendTransaction

Функция отправляет транзакцию на сервер QUIK. В случае ошибки обработки транзакции в терминале QUIK возвращает строку с диагностикой ошибки. В остальных случаях транзакция отправляется на сервер.

Результат транзакции можно получить, воспользовавшись функцией обратного вызова **OnTransReply**.

В качестве параметра принимает таблицу, в которой имена и значения полей соответствуют параметрам tri-файла (см. Руководство пользователя QUIK, Раздел 6 «Совместная работа с другими приложениями», п. 6.11.3).

**ВАЖНО! Для корректной обработки данных числовые значения (цена, количество, идентификатор транзакции и т.д.) должны передаваться в виде строковых значений.**

### 3.11.2 CalcBuySell

Функция предназначена для расчета максимально возможного количества лотов в заявке.

Формат вызова:

```
NUMBER qty, NUMBER comission CalcBuySell(String class_code, String sec_code,
String client_code, String account, NUMBER price, BOOLEAN is_buy, BOOLEAN
is_market)
```

Параметры:

- **class\_code** – код класса;
- **sec\_code** – код бумаги;
- **client\_code** – код клиента;
- **account** – счет депо;
- **price** – цена;
- **is\_buy** – признак заявки на покупку («true» – на покупку, иначе – на продажу);
- **is\_market** – признак рыночной заявки («true» – рыночная заявка, иначе – лимитированная).  
Необязательный параметр, значение по умолчанию: «false».

**При заданном параметре is\_market=true, необходимо передать параметр price=0, иначе будет рассчитано максимально возможное количество лотов в заявке по цене price.**

Пример:

```
function main()
local bs = CalcBuySell
assert(bs, "No function!!")
while not stopped do
qty, comiss = bs("BQUOTE", "AFLT", "Q3", "S01-00000F00", 10, true, false)
message("qty = " .. qty .. ", COM = " .. comiss, 2)
```



```
sleep(1000)
end
end
```

## 3.12 Функции для получения значений Таблицы текущих торгов

### 3.12.1 `getParamEx`

Функция предназначена для получения значений всех параметров биржевой информации из Таблицы текущих торгов. С помощью этой функции можно получить любое из значений Таблицы текущих торгов для заданных кодов класса и бумаги.

Формат вызова:

```
getParamEx (STRING class_code, STRING sec_code, STRING param_name)
```

Функция возвращает таблицу Lua с параметрами:

| Параметр | Тип | Описание |
|----------|-----|----------|
|----------|-----|----------|

|            |        |  |
|------------|--------|--|
| param_type | STRING | Тип данных параметра, используемый в Таблице текущих торгов. Возможные значения: |
|------------|--------|--|

- «1» – DOUBLE;
- «2» – LONG;
- «3» – CHAR;
- «4» – перечислимый тип;
- «5» – время;
- «6» – дата

|             |        |  |
|-------------|--------|--|
| param_value | STRING | Значение параметра. Для param_type = 3 значение параметра равно «0», в остальных случаях – числовое представление. Для перечислимых типов значение равно порядковому значению перечисления |
|-------------|--------|--|

|             |        |   |
|-------------|--------|---|
| param_image | STRING | Строковое значение параметра, аналогичное его представлению в таблице. В строковом представлении учитываются разделители разрядов, разделители целой и дробной части. Для перечислимых типов выводятся соответствующие им строковые значения. |
|-------------|--------|---|

### 3.12.2 `getParamEx2`

Функция предназначена для получения значений всех параметров биржевой информации из Таблицы текущих торгов с возможностью в дальнейшем отказаться от получения определенных параметров, заказанных с помощью функции [ParamRequest](#). Для отказа от получения какого-либо параметра воспользуйтесь функцией [CancelParamRequest](#).

Формат вызова:

```
getParamEx2 (STRING class_code, STRING sec_code, STRING param_name)
```

Параметры:

- **class\_code** – код класса;
- **sec\_code** – код бумаги;
- **param\_name** – код параметра.

Функция возвращает таблицу Lua с параметрами, аналогичными параметрам, возвращаемым функцией `getParamEx` (см. п. [3.12.1](#)).

## 3.13 Функции для получения параметров таблицы «Клиентский портфель»

### 3.13.1 `getPortfolioInfo`

Функция предназначена для получения значений параметров таблицы «Клиентский портфель», соответствующих идентификатору участника торгов «`firmid`» и коду клиента «`client_code`».

Формат вызова:

```
TABLE getPortfolioInfo (STRING firm_id, STRING client_code)
```

Функция возвращает таблицу Lua с параметрами:

| № | Параметр                 | Тип    | Описание  |
|---|--------------------------|--------|---|
| 1 | <code>is_leverage</code> | STRING | Признак использования схемы кредитования с контролем текущей стоимости активов.<br>Тип клиента<br>Возможные значения: <ul style="list-style-type: none"><li>– «МЛ» – используется схема ведения позиции «по плечу», «плечо» рассчитано по значению Входящего лимита,</li><li>– «МП» – используется схема ведения позиции «по плечу», «плечо» указано явным образом,</li><li>– «МОП» – используется схема ведения позиции «лимит на открытую позицию»;</li><li>– «МД» – используется схема ведения позиции «по дисконтам»;</li><li>– &lt;пусто&gt; – используется схема ведения позиции «по лимитам»</li></ul> |
| 2 | <code>in_assets</code>   | STRING | Оценка собственных средств клиента до начала торгов<br>Вход. активы   |

| №  | Параметр          | Тип    | Описание   |                    |
|----|-------------------|--------|--|--------------------|
| 3  | leverage          | STRING | Плечо. Если не задано явно, то отношение Входящего лимита к Входящим активам   | Плечо              |
| 4  | open_limit        | STRING | Оценка максимальной величины заемных средств до начала торгов  | Вход. лимит        |
| 5  | val_short         | STRING | Оценка стоимости коротких позиций. Значение всегда отрицательное   | Шорты              |
| 6  | val_long          | STRING | Оценка стоимости длинных позиций   | Лонги              |
| 7  | val_long_margin   | STRING | Оценка стоимости длинных позиций по маржинальным бумагам, принимаемым в обеспечение  | Лонги МО           |
| 8  | val_long_asset    | STRING | Оценка стоимости длинных позиций по немаржинальным бумагам, принимаемым в обеспечение  | Лонги О            |
| 9  | assets            | STRING | Оценка собственных средств клиента по текущим позициям и ценам   | Тек. активы        |
| 10 | cur_leverage      | STRING | Текущее плечо  | Тек.Плечо          |
| 11 | margin            | STRING | Уровень маржи, в процентах   | Ур. Маржи          |
| 12 | lim_all           | STRING | Текущая оценка максимальной величины заемных средств   | Тек. Лимит         |
| 13 | av_lim_all        | STRING | Оценка величины заемных средств, доступных для дальнейшего открытия позиций  | ДостТекЛимит       |
| 14 | locked_buy        | STRING | Оценка стоимости активов в заявках на покупку  | Блок. покупка      |
| 15 | locked_buy_margin | STRING | Оценка стоимости активов в заявках на покупку маржинальных бумаг, принимаемых в обеспечение                                  | Блок. пок. маржин. |
| 16 | locked_buy_asset  | STRING | Оценка стоимости активов в заявках на покупку немаржинальных бумаг, принимаемых в обеспечение                                | Блок.пок. обесп.   |
| 17 | locked_sell       | STRING | Оценка стоимости активов в заявках на продажу маржинальных бумаг   | Блок. продажа      |
| 18 | locked_value_coef | STRING | Оценка стоимости активов в заявках на покупку немаржинальных бумаг   | Блок. пок. немарж. |
| 19 | in_all_assets     | STRING | Оценка стоимости всех позиций клиента в ценах закрытия предыдущей торговой сессии, включая позиции по немаржинальным бумагам | ВходСредства       |

| №  | Параметр           | Тип    | Описание  |                     |
|----|--------------------|--------|---|---------------------|
| 20 | all_assets         | STRING | Текущая оценка стоимости всех позиций клиента   | ТекСредства         |
| 21 | profit_loss        | STRING | Абсолютная величина изменения стоимости всех позиций клиента  | Прибыль/убытки      |
| 22 | rate_change        | STRING | Относительная величина изменения стоимости всех позиций клиента   | ПроцИзмен           |
| 23 | lim_buy            | STRING | Оценка денежных средств, доступных для покупки маржинальных бумаг   | На покупку          |
| 24 | lim_sell           | STRING | Оценка стоимости маржинальных бумаг, доступных для продажи  | На продажу          |
| 25 | lim_non_margin     | STRING | Оценка денежных средств, доступных для покупки немаржинальных бумаг   | НаПокупНеМаржин     |
| 26 | lim_buy_asset      | STRING | Оценка денежных средств, доступных для покупки бумаг, принимаемых в обеспечение   | НаПокупОбесп        |
| 27 | val_short_net      | STRING | Оценка стоимости коротких позиций. При расчете не используется коэффициент дисконтирования*   | Шорты (нетто)       |
| 28 | val_long_net       | STRING | Оценка стоимости длинных позиций. При расчете не используется коэффициент дисконтирования*  | Лонги (нетто)       |
| 29 | total_money_bal    | STRING | Сумма остатков по денежным средствам по всем лимитам, без учета средств, заблокированных под исполнение обязательств, выраженная в выбранной валюте расчета   | Сумма ден. остатков |
| 30 | total_locked_money | STRING | Сумма заблокированных средств со всех денежных лимитов клиента, пересчитанная в валюту расчетов через кросс-курсы на сервере  | Суммарно заблок.    |
| 31 | haircuts           | STRING | Сумма дисконтов стоимости длинных (только по бумагам обеспечения) и коротких бумажных позиций, дисконтов корреляции между инструментами, а также дисконтов на задолженности по валютам, не покрытые бумажным обеспечением в этих же валютах                                       | Сумма дисконтов     |
| 32 | assets_without_hc  | STRING | Суммарная величина денежных остатков, стоимости длинных позиций по бумагам обеспечения и стоимости коротких позиций, без учета дисконтирующих коэффициентов, без учета неттинга стоимости бумаг в рамках объединенной бумажной позиции и без учета корреляции между инструментами | ТекАктБезДиск       |
| 33 | status_coef        | STRING | Отношение суммы дисконтов к текущим активам без учета дисконтов   | Статус счета        |

| №  | Параметр         | Тип    | Описание  |                    |
|----|------------------|--------|---|--------------------|
| 34 | varmargin        | STRING | Текущая вариационная маржа по позициям клиента, по всем инструментам  | Вариационная маржа |
| 35 | go_for_positions | STRING | Размер денежных средств, уплаченных под все открытые позиции на срочном рынке   | ГО поз.            |
| 36 | go_for_orders    | STRING | Оценка стоимости активов в заявках на срочном рынке   | ГО заяв.           |
| 37 | rate_futures     | STRING | Отношение ликвидационной стоимости портфеля к ГО по срочному рынку  | Активы/ГО          |
| 38 | is_qual_client   | STRING | Признак «квалифицированного» клиента, которому разрешено кредитование заемными средствами с плечом 1:3. Возможные значения: «ПовышУрРиска» – квалифицированный, <пусто> – нет | ПовышУрРиска       |
| 39 | is_futures       | STRING | Счет клиента на FORTS, в случае наличия объединенной позиции, иначе поле остается пустым  | Сроч. счет         |
| 40 | curr_TAG         | STRING | Актуальные текущие параметры расчета для данной строки в формате «<Валюта>-<Идентификатор торговой сессии>». Пример: «SUR-EQTV»   | Парам. расч.       |

**(\*) Подробнее о коэффициентах дисконтирования см. п. 7 Руководства по администрированию «Настройки Библиотеки расчета лимитов».**

### 3.13.2 getPortfolioInfoEx

Функция предназначена для получения значений параметров таблицы «Клиентский портфель», соответствующих идентификатору участника торгов «firmid», коду клиента «client\_code» и виду лимита «limit\_kind».

Формат вызова:

```
TABLE getPortfolioInfoEx (STRING firm_id, STRING client_code, NUMBER limit_kind)
```

Функция возвращает таблицу Lua с параметрами таблицы «Клиентский портфель». Описание параметров см. в п. [3.13.1](#).

Дополнительно возвращаются следующие параметры:

| № | Параметр         | Тип    | Описание  |
|---|------------------|--------|---|
| 1 | init_margin      | STRING | Значение начальной маржи. Заполняется для клиентов типа «МД»  |
| 2 | min_margin       | STRING | Значение минимальной маржи. Заполняется для клиентов типа «МД»  |
| 3 | corrected_margin | STRING | Значение скорректированной маржи. Заполняется для клиентов типа «МД»  |
| 4 | client_type      | STRING | Тип клиента   |
| 5 | portfolio_value  | STRING | Стоимость портфеля. Для клиентов типа «МД» возвращается значение для строк с максимальным видом лимита limit_kind |

## 3.14 Функции для получения параметров таблицы «Купить/Продать»

### 3.14.1 getBuySellInfo

Функция предназначена для получения значений параметров таблицы «Купить/Продать».

Формат вызова:

```
TABLE getBuySellInfo (STRING firm_id, STRING client_code, STRING class_code, STRING sec_code, NUMBER price)
```

Функция возвращает таблицу Lua с параметрами из таблицы QUIK «Купить/Продать», означающими возможность купить либо продать указанный инструмент «sec\_code» класса «class\_code», указанным клиентом «client\_code» фирмы «firmid», по указанной цене «price». Если цена равна «0», то используются лучшие значения спроса/предложения.

Параметры:

| № | Параметр      | Тип    | Описание  |
|---|---------------|--------|---|
| 1 | is_margin_sec | STRING | Признак маржинальности инструмента. Возможные значения:<br>_ «0» – не маржинальная;<br>_ «1» – маржинальная |

| №  | Параметр           | Тип    | Описание  |
|----|--------------------|--------|---|
| 2  | is_asset_sec       | STRING | Принадлежность инструмента к списку бумаг, принимаемых в обеспечение. Возможные значения:<br>_ «0» – не принимается в обеспечение;<br>_ «1» – принимается в обеспечение |
| 3  | balance            | STRING | Текущая позиция по инструменту, в лотах   |
| 4  | can_buy            | STRING | Оценка количества лотов, доступных на покупку по указанной цене *   |
| 5  | can_sell           | STRING | Оценка количества лотов, доступных на продажу по указанной цене *   |
| 6  | position_valuation | STRING | Денежная оценка позиции по инструменту по ценам спроса/предложения  |
| 7  | value              | STRING | Оценка стоимости позиции по цене последней сделки   |
| 8  | open_value         | STRING | Оценка стоимости позиции клиента, рассчитанная по цене закрытия предыдущей торговой сессии  |
| 9  | lim_long           | STRING | Предельный размер позиции по данному инструменту, принимаемый в обеспечение длинных позиций   |
| 10 | long_coef          | STRING | Коэффициент дисконтирования, применяемый для длинных позиций по данному инструменту   |
| 11 | lim_short          | STRING | Предельный размер короткой позиции по данному инструменту   |
| 12 | short_coef         | STRING | Коэффициент дисконтирования, применяемый для коротких позиций по данному инструменту  |
| 13 | value_coef         | STRING | Оценка стоимости позиции по цене последней сделки, с учетом дисконтирующих коэффициентов  |
| 14 | open_value_coef    | STRING | Оценка стоимости позиции клиента, рассчитанная по цене закрытия предыдущей торговой сессии с учетом дисконтирующих коэффициентов  |
| 15 | share              | STRING | Процентное отношение стоимости позиции по данному инструменту к стоимости всех активов клиента, рассчитанное по текущим ценам   |
| 16 | short_wa_price     | STRING | Средневзвешенная стоимость коротких позиций по инструментам   |
| 17 | long_wa_price      | STRING | Средневзвешенная стоимость длинных позиций по инструментам  |
| 18 | profit_loss        | STRING | Разница между средневзвешенной ценой приобретения бумаг и их рыночной оценки  |
| 19 | spread_hc          | STRING | Коэффициент корреляции между инструментами  |
| 20 | can_buy_own        | STRING | Максимально возможное количество бумаг в заявке на покупку этого инструмента на этом классе на собственные средства клиента, исходя из цены лучшего предложения         |

| №  | Параметр     | Тип    | Описание  |
|----|--------------|--------|---|
| 21 | can_sell_own | STRING | Максимально возможное количество бумаг в заявке на продажу этого инструмента на этом классе из собственных активов клиента, исходя из цены лучшего спроса |

**(\*) В зависимости от настроек сервера QUIK, величина может выражаться в лотах или в штуках. Уточните единицы измерения у обслуживающего брокера.**

### 3.14.2 getBuySellInfoEx

Функция предназначена для получения значений параметров таблицы «Купить/Продать».

Формат вызова:

```
TABLE getBuySellInfoEx (STRING firm_id, STRING client_code, STRING class_code,
STRING sec_code, NUMBER price)
```

Функция возвращает таблицу Lua с параметрами из таблицы QUIK «Купить/Продать», означающими возможность купить либо продать указанный инструмент «sec\_code» класса «class\_code», указанным клиентом «client\_code» фирмы «firmid», по указанной цене «price». Если цена равна «0», то используются лучшие значения спроса/предложения.

Описание возвращаемых параметров см. в п. [3.14.1](#).

Дополнительно возвращаются следующие параметры:

| № | Параметр    | Тип    | Описание  |
|---|-------------|--------|---|
| 1 | limit_kind  | NUMBER | Вид лимита. Возможные значения:<br>_ «0» – «T0»;<br>_ «1» – «T1»;<br>_ «2» – «T2»           |
| 2 | d_long      | STRING | Эффективный начальный дисконт для длинной позиции.<br>Заполняется для клиентов типа «МД»    |
| 3 | d_min_long  | STRING | Эффективный минимальный дисконт для длинной позиции.<br>Заполняется для клиентов типа «МД»  |
| 4 | d_short     | STRING | Эффективный начальный дисконт для короткой позиции.<br>Заполняется для клиентов типа «МД»   |
| 5 | d_min_short | STRING | Эффективный минимальный дисконт для короткой позиции.<br>Заполняется для клиентов типа «МД» |



| № | Параметр         | Тип    | Описание  |
|---|------------------|--------|---|
| 6 | client_type      | STRING | Тип клиента. Возможные значения:<br>— «1» – «МЛ»;<br>— «2» – «МП»;<br>— «3» – «МОП»;<br>— «4» – «МД»  |
| 7 | is_long_allowed  | STRING | Признак того, является ли бумага разрешенной для покупки на заемные средства. Возможные значения:<br>«1» – разрешена, «0» – не разрешена.<br>Заполняется для клиентов типа «МД» |
| 8 | is_short_allowed | STRING | Признак того, является ли бумага разрешенной для продажи на заемные средства. Возможные значения:<br>«1» – разрешена, «0» – не разрешена.<br>Заполняется для клиентов типа «МД» |

## 3.15 Функции для работы с таблицами Рабочего места QUIK

В таблицах Рабочего места QUIK, созданных с помощью скриптов на языке Lua, поддерживаны следующие возможности:

- режим «drag-and-drop»;
- пользовательские фильтры;
- условное форматирование;
- размещение на «экранных закладках»;
- поиск значения в ячейках таблицы;
- печать таблицы с предварительным просмотром.

Ниже приведен список действий, которые не поддерживаются для таблиц, созданных на языке Lua:

- таблицы не сохраняются в файл конфигурации;
- отсутствует диалог редактирования таблицы;
- отсутствует контекстное меню таблицы (кроме пункта «Переместить на закладку»);
- не создается копия таблицы;
- не задается по умолчанию заголовков окна таблицы;
- отсутствует экспорт данных из таблицы;
- недоступно управление «горячими клавишами».

### 3.15.1 AddColumn

Функция добавляет колонки в таблицу с идентификатором «t\_id».

Формат вызова:

```
NUMBER AddColumn (NUMBER t_id, NUMBER iCode, STRING name, BOOLEAN is_default,  
NUMBER par_type, NUMBER width)
```

Параметры:

- **iCode** – код параметра, выводимого в колонке;
- **name** – название колонки;
- **is\_default** – параметр не используется;
- **par\_type** – тип данных в колонке, одна из следующих констант:
  - \_ QTABLE\_INT\_TYPE – целое число;
  - \_ QTABLE\_DOUBLE\_TYPE – число с плавающей точкой;
  - \_ QTABLE\_INT64\_TYPE – 64-битное целое число;
  - \_ QTABLE\_CACHED\_STRING\_TYPE – кэшируемая строка;
  - \_ QTABLE\_TIME\_TYPE – время;
  - \_ QTABLE\_DATE\_TYPE – дата;
  - \_ QTABLE\_STRING\_TYPE – строка.
- **width** – ширина в условных единицах.

Функция возвращает «1», если колонка в таблицу добавлена, иначе – «0».

### 3.15.2 AllocTable

Функция создает структуру, описывающую таблицу.

Формат вызова:

```
NUMBER AllocTable()
```

Функция возвращает целочисленный идентификатор таблицы, предназначенный для совершения с ней дальнейших операций.

### 3.15.3 Clear

Функция удаляет содержимое таблицы с идентификатором «t\_id».

Формат вызова:

```
NUMBER Clear (NUMBER t_id)
```

Функция возвращает «false» в случае неуспешного завершения.

### 3.15.4 CreateWindow

Функция создает окно таблицы с идентификатором «t\_id».

Формат вызова:

```
NUMBER CreateWindow(NUMBER t_id)
```

Функция возвращает «1» при успешном создании окна, иначе – «0».

### 3.15.5 DeleteRow

Функция удаления строки с ключом «key» из таблицы с идентификатором «t\_id».

Формат вызова:

```
BOOLEAN DeleteRow(NUMBER t_id, NUMBER key)
```

Функция возвращает «false» в случае неуспешного завершения.

### 3.15.6 DestroyTable

Функция закрывает окно таблицы с идентификатором «t\_id».

Формат вызова:

```
NUMBER DestroyTable(NUMBER t_id)
```

Все данные для отображения при закрытии окна удаляются.

В случае успешного выполнения возвращает «true», иначе – «false».

### 3.15.7 InsertRow

Функция добавляет строку с ключом «key» в таблицу с идентификатором «t\_id».

Формат вызова:

```
NUMBER InsertRow(NUMBER t_id, NUMBER key)
```

**При добавлении данных в новую таблицу в первую очередь выполните данную функцию с параметром «key» равным «-1». При этом строка добавится в конец таблицы.**

Функция возвращает номер добавленной строки при успешном выполнении, иначе – «-1».

При вызове функции с параметром «key» большим, чем текущее количество строк, строка добавляется в конец таблицы.

### 3.15.8 IsWindowClosed

Функция возвращает «true», если окно с таблицей «t\_id» закрыто.

**Вызов IsWindowClosed внутри функции обратного вызова, заданной с помощью SetTableNotificationCallback(), всегда вернет «false».**

Формат вызова:

BOOLEAN IsWindowClosed(NUMBER t\_id)

Функция возвращает «nil» в случае неуспешного завершения.

Окно может быть открыто повторно с помощью функции **CreateWindow** (подробнее см. п. [3.15.4](#)).

### 3.15.9 GetCell

Функция возвращает таблицу, содержащую данные из ячейки в строке с ключом «key», кодом колонки «code» в таблице «t\_id».

Формат вызова:

TABLE GetCell(NUMBER t\_id, NUMBER key, NUMBER code)

Параметры таблицы:

- **image** – строковое представление значения в ячейке,
- **value** – числовое значение ячейки.

Если входные параметры были заданы ошибочно, то возвращается «nil».

### 3.15.10 GetTableSize

Функция возвращает количество строк «rows» и столбцов «col» в таблице с индексом «t\_id».

Формат вызова:

NUMBER rows, NUMBER col GetTableSize (NUMBER t\_id)

Пользовательские фильтры, установленные на таблице, не влияют на возвращаемое количество строк. Заголовки строк и первый фиксированный столбец не учитываются в возвращаемых значениях.

Функция возвращает «nil» в случае неуспешного завершения.

### 3.15.11 GetWindowCaption

Функция получает текущий заголовок окна.

Формат вызова:

STRING GetWindowCaption(NUMBER t\_id)

Функция возвращает «nil» в случае неуспешного завершения.

### 3.15.12 GetWindowRect

Функция возвращает координаты верхнего левого и нижнего правого угла окна, содержащего таблицу «t\_id».

Формат вызова:

```
NUMBER top, NUMBER left, NUMBER bottom, NUMBER right  
GetWindowRect(NUMBER t_id)
```

Функция возвращает «nil» в случае неуспешного завершения.

### 3.15.13 SetCell

Функция задает значение для ячейки в строке с ключом «key», кодом колонки «code» в таблице «t\_id».

Формат вызова:

```
BOOLEAN SetCell(NUMBER t_id, NUMBER key, NUMBER code, STRING text,  
NUMBER value)
```

Параметр «text» задает строковое представление значения параметра «value». Параметр «value» необязательный и по умолчанию равен «0». Для столбцов со строковыми типами данных параметр «value» не задается.

Если параметр «value» не задан для ячеек всех остальных типов, то по столбцам, содержащим такие ячейки, не будет корректно работать сортировка, фильтрация и условное форматирование (см. [Приложение 2](#)).

Функция возвращает «true» в случае успешного завершения, иначе – «false».

### 3.15.14 SetWindowCaption

Функция задает новый заголовок окна.

Формат вызова:

```
SetWindowCaption(NUMBER t_id, STRING str)
```

В случае успешного выполнения возвращает «true», иначе – «false».

### 3.15.15 SetWindowPos

Функция устанавливает положение окна с таблицей «t\_id». Левый верхний угол в координаты x,y и размеры в dx, dy.

Формат вызова:

```
BOOLEAN SetWindowPos(NUMBER t_id, NUMBER x, NUMBER y, NUMBER dx,  
NUMBER dy)
```

В случае успешного выполнения возвращает «true», иначе – «false».

### 3.15.16 SetTableNotificationCallback

Задание функции обратного вызова для обработки событий в таблице.

Формат вызова:

```
NUMBER SetTableNotificationCallback (NUMBER t_id, FUNCTION f_cb)
```

Параметры:

- **t\_id** – идентификатор таблицы,
- **f\_cb** – функция обратного вызова для обработки событий в таблице.

В случае успешного завершения функция возвращает «1», иначе – «0».

Формат вызова функции обратного вызова для обработки событий в таблице:

```
FUNCTION (NUMBER t_id, NUMBER msg, NUMBER par1, NUMBER par2)
```

Параметры:

- **t\_id** – идентификатор таблицы, для которой обрабатывается сообщение,
- **par1** и **par2** – значения параметров определяются типом сообщения msg,
- **msg** – код сообщения.

Доступные коды событий:

- **QTABLE\_LBUTTONDOWN** – нажата левая кнопка мыши, при этом par1 содержит номер строки, par2 – номер колонки,
- **QTABLE\_RBUTTONDOWN** – нажата правая кнопка мыши, при этом par1 содержит номер строки, par2 – номер колонки,
- **QTABLE\_LBUTTONDBLCLK** – двойное нажатие левой кнопки мыши, при этом par1 содержит номер строки, par2 – номер колонки,
- **QTABLE\_RBUTTONDBLCLK** – двойное нажатие правой кнопки мыши, при этом par1 содержит номер строки, par2 – номер колонки,
- **QTABLE\_SELCHANGED** – изменение текущей (выделенной) строки, при этом par1 равен номеру новой выделенной строки,
- **QTABLE\_CHAR** – нажата символьная клавиша, при этом par2 содержит код клавиши, par1 – текущую выделенную строку,
- **QTABLE\_VKEY** – нажата клавиша, при этом par2 содержит код клавиши, par1 – текущую выделенную строку,
- **QTABLE\_MBUTTONDOWN** – нажата вниз средняя кнопка мыши, при этом par1 содержит номер строки, par2 – номер колонки,

- `QTABLE_MBUTTONDOWNBLCLK` – двойное нажатие средней кнопки мыши, при этом `par1` содержит номер строки, `par2` – номер колонки,
- `QTABLE_LBUTTONUP` – отпущена левая кнопка мыши, при этом `par1` содержит номер строки, `par2` – номер колонки,
- `QTABLE_RBUTTONUP` – отпущена правая кнопка мыши, при этом `par1` содержит номер строки, `par2` – номер колонки,
- `QTABLE_CLOSE` – закрытие таблицы, при этом `par1` и `par2` равны нулю.

### 3.15.17 RGB

Функция преобразовывает компоненты RGB (red, green, blue) в одно число для дальнейшего использования в функции `SetColor` (см. п. [3.15.18](#)).

Формат вызова:

```
NUMBER RGB(NUMBER red, NUMBER green, NUMBER blue)
```

### 3.15.18 SetColor

Функция устанавливает цвет ячейки, столбца или строки для таблицы с идентификатором «`t_id`».

Формат вызова:

```
BOOLEAN SetColor(NUMBER t_id, NUMBER row, NUMBER col, NUMBER b_color,
NUMBER f_color, NUMBER sel_b_color, NUMBER sel_f_color)
```

Параметры, являющиеся результатом выполнения функции RGB:

- **b\_color** – цвет фона,
- **f\_color** – цвет текста,
- **sel\_b\_color** – цвет фона выделенной ячейки,
- **sel\_f\_color** – цвет текста выделенной ячейки.

В зависимости от переданных параметров **row** и **col** можно менять цвет всей таблицы, столбца, строки и отдельной ячейки.

Если в качестве цвета задана константа `QTABLE_DEFAULT_COLOR`, то используется цвет, заданный в цветовой схеме операционной системе Windows.

При работе функции используется константа `QTABLE_NO_INDEX`, равная «-1».

Ниже приведены варианты установки цвета в таблице:

| row                        | col                           | Результат                         |
|----------------------------|-------------------------------|-----------------------------------|
| Количество строк от 1 до N | Количество столбцов от 1 до M | Ячейка раскрашена заданным цветом |

| row                        | col                           | Результат                                    |
|----------------------------|-------------------------------|--|
| Количество строк от 1 до N | QTABLE_NO_INDEX               | Строка раскрашена заданным цветом            |
| QTABLE_NO_INDEX            | Количество столбцов от 1 до M | Столбец раскрашен заданным цветом            |
| QTABLE_NO_INDEX            | QTABLE_NO_INDEX               | Заданным цветом раскрашена полностью таблица |

### 3.15.19 Highlight

Функция для подсветки выбранного диапазона ячеек цветом фона и цветом текста в течении указанного времени с плавным «затуханием» для таблицы с идентификатором «t\_id».

Формат вызова:

```
BOOLEAN Highlight(NUMBER t_id, NUMBER row, NUMBER col, NUMBER b_color,
NUMBER f_color, NUMBER timeout)
```

Параметры:

- **b\_color** – цвет фона,
- **f\_color** – цвет текста,
- **timeout** – время подсветки, в миллисекундах.

Для отмены подсветки вызовите функцию с параметром timeout равным «0». При этом параметры «b\_color» и «f\_color» могут содержать любые значения.

Варианты подсветки ячеек в таблице аналогичны вариантам установки цвета, приведенным в описании функции SetColor (см. п. [3.15.18](#)).

### 3.15.20 SetSelectedRow

Функция выделяет определенную строку таблицы.

```
NUMBER row SetSelectedRow((NUMBER table_id, NUMBER row)
```

Параметры:

- **table\_id** – идентификатор таблицы;
- **row** – номер строки.

**Если задано значение row=-1, то выделяется последняя видимая строка в таблице.**

В случае успешного завершения функция возвращает номер выделенной строки, иначе – «-1».



Функция работает с видимым представлением таблицы, в котором учитываются пользовательские фильтры и сортировка.

## 3.16 Функции для работы с метками

Функции предназначены для построения меток и установки их на графике.

### 3.16.1 AddLabel

Добавляет метку с заданными параметрами.

```
NUMBER AddLabel(String chart_tag, Table label_params)
```

Параметры:

- **chart\_tag** – тег графика, к которому привязывается метка (подробнее см. п. 4.2.3 в Разделе 4 Руководства пользователя QUIK, настройка «Идентификатор»);
- **label\_params** – таблица с параметрами метки.

Функция возвращает числовой идентификатор метки. В случае неуспешного завершения функция возвращает «nil».

Формат таблицы с параметрами метки:

| № | Параметр   | Тип    | Описание   |
|---|------------|--------|--|
| 1 | TEXT       | STRING | Подпись метки (если подпись не требуется, то пустая строка)  |
| 2 | IMAGE_PATH | STRING | Путь к картинке, которая будет отображаться в качестве метки (пустая строка, если картинка не требуется) |
| 3 | ALIGNMENT  | STRING | Расположение картинки относительно текста (возможно 4 варианта: LEFT, RIGHT, TOP, BOTTOM)                |
| 4 | YVALUE     | NUMBER | Значение параметра на оси Y, к которому будет привязана метка  |
| 5 | DATE       | NUMBER | Дата в формате «ГГГГММДД», к которой привязана метка   |
| 6 | TIME       | NUMBER | Время в формате «ЧЧММСС», к которому будет привязана метка   |
| 7 | R          | NUMBER | Красная компонента цвета в формате RGB. Число в интервале [0;255]  |

| №  | Параметр               | Тип    | Описание  |
|----|------------------------|--------|---|
| 8  | G                      | NUMBER | Зеленая компонента цвета в формате RGB. Число в интервале [0;255]                                 |
| 9  | B                      | NUMBER | Синяя компонента цвета в формате RGB. Число в интервале [0;255]                                   |
| 10 | TRANSPARENCY           | NUMBER | Прозрачность метки в процентах. Значение должно быть в промежутке [0; 100]                        |
| 11 | TRANSPARENT_BACKGROUND | NUMBER | Прозрачность метки. Возможные значения: «0» – прозрачность отключена, «1» – прозрачность включена |
| 12 | FONT_FACE_NAME         | STRING | Название шрифта (например «Arial»)  |
| 13 | FONT_HEIGHT            | NUMBER | Размер шрифта   |
| 14 | HINT                   | STRING | Текст всплывающей подсказки   |

### 3.16.2 DelLabel

Удаляет метку с заданными параметрами.

```
BOOLEAN DelLabel(String chart_tag, NUMBER label_id)
```

Параметры:

- **chart\_tag** – тег графика, к которому привязывается метка,
- **label\_id** – идентификатор метки.

В случае успешного завершения функция возвращает «true», иначе – «false».

### 3.16.3 DelAllLabels

Команда удаляет все метки на диаграмме с указанным графиком.

```
BOOLEAN DelAllLabels(String chart_tag)
```

Параметры:

- **chart\_tag** – тег графика, к которому привязывается метка.

В случае успешного завершения функция возвращает «true», иначе – «false».

### 3.16.4 GetLabelParams

Команда позволяет получить параметры метки.

```
TABLE GetLabelParams(String chart_tag, NUMBER label_id)
```

Параметры:

- **chart\_tag** – тег графика, к которому привязывается метка,
- **label\_id** – идентификатор метки.

Функция возвращает таблицу с параметрами метки. В случае неуспешного завершения функция возвращает «nil».

**Наименование параметров метки в возвращаемой таблице указаны в нижнем регистре, и все значения имеют тип – STRING.**

### 3.16.5 SetLabelParams

Функция задает параметры для метки с указанным идентификатором.

```
BOOLEAN SetLabelParams(STRING chart_tag, NUMBER label_id, TABLE label_params)
```

Параметры:

- **chart\_tag** – тег графика, к которому привязывается метка,
- **label\_id** – идентификатор метки,
- **label\_params** – таблица с новыми параметрами метки.

В случае успешного завершения функция возвращает «true», иначе – «false».

## 3.17 Функции для заказа стакана котировок

### 3.17.1 Subscribe\_Level\_II\_Quotes

Функция заказывает на сервер получение стакана по указанному классу и бумаге.

```
BOOLEAN Subscribe_Level_II_Quotes(STRING class_code, STRING sec_code)
```

Параметры:

- **class\_code** – код класса,
- **sec\_code** – код бумаги

В случае успешного завершения функция возвращает «true».

### 3.17.2 Unsubscribe\_Level\_II\_Quotes

Функция отменяет заказ на получение с сервера стакана по указанному классу и бумаге.

```
BOOLEAN Unsubscribe_Level_II_Quotes(STRING class_code, STRING sec_code)
```

Параметры:

- **class\_code** – код класса,
- **sec\_code** – код бумаги.

В случае успешного завершения функция возвращает «true».

### 3.17.3 IsSubscribed\_Level\_II\_Quotes

Функция позволяет узнать, заказан ли с сервера стакан по указанному классу и бумаге.

BOOLEAN IsSubscribed\_Level\_II\_Quotes (STRING class\_code, STRING sec\_code)

Параметры:

- **class\_code** – код класса,
- **sec\_code** – код бумаги

Функция возвращает «true», если стакан по классу **class\_code** и бумаге **sec\_code** уже заказан.

## 3.18 Функции для заказа параметров Таблицы текущих торгов

### 3.18.1 ParamRequest

Функция заказывает получение параметров Таблицы текущих торгов.

BOOLEAN ParamRequest(STRING class\_code, STRING sec\_code, STRING db\_name)

**Для корректной работы функции включите в настройках Рабочего места QUIK признак получения данных «Исходя из настроек открытых пользователем таблиц» (меню Система / Настройки / Основные настройки..., раздел «Программа» / «Получение данных»).**

Параметры:

- **class\_code** – код класса;
- **sec\_code** – код бумаги;
- **db\_name** – код параметра.

В случае успешного завершения функция возвращает «true», иначе – «false».

### 3.18.2 CancelParamRequest

Функция отменяет заказ на получение параметров Таблицы текущих торгов.

BOOLEAN CancelParamRequest(String class\_code, STRING sec\_code, STRING db\_name)

Для корректной работы функции включите в настройках Рабочего места QUIK признак получения данных «Исходя из настроек открытых пользователем таблиц» (меню Система / Настройки / Основные настройки..., раздел «Программа» / «Получение данных»).

Параметры:

- **class\_code** – код класса;
- **sec\_code** – код бумаги;
- **db\_name** – код параметра.

В случае успешного завершения функция возвращает «true», иначе – «false».

## 4. Структуры данных

### 4.1 Классы

Описание параметров Таблицы классов:

| Параметр | Тип    | Описание                       |
|----------|--------|--------------------------------|
| firmid   | STRING | Идентификатор фирмы            |
| name     | STRING | Название класса                |
| code     | STRING | Код класса                     |
| npars    | NUMBER | Количество параметров в классе |
| nsecs    | NUMBER | Количество бумаг в классе      |

Пример:

```
t={
  ["firmid"]="NC0038900000",
  ["name"]="Брокерские котировки",
  ["code"]="BQUOTE",
  ["npars"]=38,
  ["nsecs"]=28
}
```

См. также описание функций [getItem](#), [getNumberOf](#).

## 4.2 Фирмы

Описание параметров Таблицы фирм:

| Параметр  | Тип    | Описание            |
|-----------|--------|---------------------|
| firmid    | STRING | Идентификатор фирмы |
| firm_name | STRING | Название фирмы      |
| status    | NUMBER | Статус              |
| exchange  | STRING | Торговая площадка   |

См. также описание функций [getItem](#), [getNumberOf](#).

## 4.3 Обезличенные сделки

Описание параметров Таблицы обезличенных сделок:

| Параметр   | Тип    | Описание                             |
|------------|--------|--------------------------------------|
| trade_num  | NUMBER | Идентификатор сделки                 |
| flags      | NUMBER | <a href="#">Набор битовых флагов</a> |
| price      | NUMBER | Цена                                 |
| qty        | NUMBER | Количество                           |
| value      | NUMBER | Объем сделки                         |
| accruedint | NUMBER | Накопленный купонный доход           |
| yield      | NUMBER | Доходность                           |
| settlecode | STRING | Код расчетов                         |
| reporate   | NUMBER | Ставка РЕПО                          |
| repovalue  | NUMBER | Сумма РЕПО                           |
| repo2value | NUMBER | Объем сделки выкупа РЕПО             |
| repoterm   | NUMBER | Срок РЕПО в днях                     |
| sec_code   | STRING | Код инструмента                      |
| class_code | STRING | Код класса                           |

| Параметр      | Тип    | Описание   |
|---------------|--------|--|
| datetime      | TABLE  | <a href="#">Дата и время</a>   |
| period        | NUMBER | Период торговой сессии. Возможные значения: <ul style="list-style-type: none"> <li>«0» – Открытие;</li> <li>«1» – Нормальный;</li> <li>«2» – Закрытие</li> </ul> |
| open_interest | NUMBER | Открытый интерес   |

## 4.4 Сделки

Описание параметров Таблицы сделок:

| Параметр    | Тип    | Описание   |
|-------------|--------|--|
| trade_num   | NUMBER | Номер сделки в торговой системе                      |
| order_num   | NUMBER | Номер заявки в торговой системе                      |
| brokerref   | STRING | Комментарий, обычно: <код клиента>/<номер поручения> |
| userid      | STRING | Идентификатор трейдера                               |
| firmid      | STRING | Идентификатор дилера                                 |
| account     | STRING | Торговый счет  |
| price       | NUMBER | Цена   |
| qty         | NUMBER | Количество бумаг в последней сделке в лотах          |
| value       | NUMBER | Объем в денежных средствах                           |
| accruedint  | NUMBER | Накопленный купонный доход                           |
| yield       | NUMBER | Доходность   |
| settlecode  | STRING | Код расчетов   |
| cpfirmid    | STRING | Код фирмы партнера                                   |
| flags       | NUMBER | <a href="#">Набор битовых флагов</a>                 |
| price2      | NUMBER | Цена выкупа  |
| reporate    | NUMBER | Ставка РЕПО (%)                                      |
| client_code | STRING | Код клиента  |
| accrued2    | NUMBER | Доход (%) на дату выкупа                             |
| repoterm    | NUMBER | Срок РЕПО, в календарных днях                        |

| Параметр              | Тип    | Описание   |
|-----------------------|--------|--|
| repovalue             | NUMBER | Сумма РЕПО   |
| repo2value            | NUMBER | Объем выкупа РЕПО  |
| start_discount        | NUMBER | Начальный дисконт (%)  |
| lower_discount        | NUMBER | Нижний дисконт (%)   |
| upper_discount        | NUMBER | Верхний дисконт (%)  |
| block_securities      | NUMBER | Блокировка обеспечения («Да»/«Нет»)  |
| clearing_comission    | NUMBER | Клиринговая комиссия биржи   |
| exchange_comission    | NUMBER | Комиссия Фондовой биржи  |
| tech_center_comission | NUMBER | Комиссия Технического центра   |
| settle_date           | NUMBER | Дата расчетов  |
| settle_currency       | STRING | Валюта расчетов  |
| trade_currency        | STRING | Валюта   |
| exchange_code         | STRING | Код биржи в торговой системе   |
| station_id            | STRING | Идентификатор рабочей станции  |
| sec_code              | STRING | Код бумаги заявки  |
| class_code            | STRING | Код класса   |
| datetime              | TABLE  | <a href="#">Дата и время</a>   |
| bank_acc_id           | STRING | Идентификатор расчетного счета/кода в клиринговой организации  |
| broker_comission      | NUMBER | Комиссия брокера. Отображается с точностью до 2 двух знаков. Поле зарезервировано для будущего использования   |
| linked_trade          | NUMBER | Номер витринной сделки в Торговой Системе для сделок РЕПО с ЦК и SWAP  |
| period                | NUMBER | Период торговой сессии. Возможные значения: <ul style="list-style-type: none"> <li>_ «0» – Открытие;</li> <li>_ «1» – Нормальный;</li> <li>_ «2» – Закрытие</li> </ul> |
| trans_id              | NUMBER | Идентификатор транзакции   |



| Параметр            | Тип    | Описание   |
|---------------------|--------|--|
| kind                | NUMBER | <p>Тип сделки. Возможные значения:</p> <ul style="list-style-type: none"> <li>– «1» – Обычная;</li> <li>– «2» – Адресная;</li> <li>– «3» – Первичное размещение;</li> <li>– «4» – Перевод денег/бумаг;</li> <li>– «5» – Адресная сделка первой части РЕПО;</li> <li>– «6» – Расчетная по операции своп;</li> <li>– «7» – Расчетная по внебиржевой операции своп;</li> <li>– «8» – Расчетная сделка бивалютной корзины;</li> <li>– «9» – Расчетная внебиржевая сделка бивалютной корзины;</li> <li>– «10» – Сделка по операции РЕПО с ЦК;</li> <li>– «11» – Первая часть сделки по операции РЕПО с ЦК;</li> <li>– «12» – Вторая часть сделки по операции РЕПО с ЦК;</li> <li>– «13» – Адресная сделка по операции РЕПО с ЦК;</li> <li>– «14» – Первая часть адресной сделки по операции РЕПО с ЦК;</li> <li>– «15» – Вторая часть адресной сделки по операции РЕПО с ЦК;</li> <li>– «16» – Техническая сделка по возврату активов РЕПО с ЦК;</li> <li>– «17» – Сделка по спреду между фьючерсами разных сроков на один актив;</li> <li>– «18» – Техническая сделка первой части от спреда между фьючерсами;</li> <li>– «19» – Техническая сделка второй части от спреда между фьючерсами;</li> <li>– «20» – Адресная сделка первой части РЕПО с корзиной;</li> <li>– «21» – Адресная сделка второй части РЕПО с корзиной;</li> <li>– «22» – Перенос позиций срочного рынка</li> </ul> |
| clearing_bank_accid | STRING | Идентификатор счета в НКЦ (расчетный код)  |
| canceled_datetime   | TABLE  | <a href="#">Дата и время</a> снятия сделки   |
| clearing_firmid     | STRING | Идентификатор фирмы - участника клиринга   |
| system_ref          | STRING | Дополнительная информация по сделке, передаваемая торговой системой  |
| uid                 | NUMBER | Идентификатор пользователя на сервере QUIK   |

## 4.5 Заявки

Описание параметров Таблицы заявок:

| Параметр          | Тип    | Описание   |
|-------------------|--------|--|
| order_num         | NUMBER | Номер заявки в торговой системе                      |
| * flags           | NUMBER | <a href="#">Набор битовых флагов</a>                 |
| brokerref         | STRING | Комментарий, обычно: <код клиента>/<номер поручения> |
| userid            | STRING | Идентификатор трейдера                               |
| firmid            | STRING | Идентификатор фирмы                                  |
| account           | STRING | Торговый счет  |
| price             | NUMBER | Цена   |
| qty               | NUMBER | Количество в лотах                                   |
| balance           | NUMBER | Остаток  |
| value             | NUMBER | Объем в денежных средствах                           |
| accruedint        | NUMBER | Накопленный купонный доход                           |
| yield             | NUMBER | Доходность   |
| trans_id          | NUMBER | Идентификатор транзакции                             |
| client_code       | STRING | Код клиента  |
| price2            | NUMBER | Цена выкупа  |
| settlecode        | STRING | Код расчетов   |
| uid               | NUMBER | Идентификатор пользователя                           |
| exchange_code     | STRING | Код биржи в торговой системе                         |
| activation_time   | NUMBER | Время активации                                      |
| linkedorder       | NUMBER | Номер заявки в торговой системе                      |
| expiry            | NUMBER | Дата окончания срока действия заявки                 |
| sec_code          | STRING | Код бумаги заявки                                    |
| class_code        | STRING | Код класса заявки                                    |
| datetime          | TABLE  | <a href="#">Дата и время</a>                         |
| withdraw_datetime | TABLE  | <a href="#">Дата и время</a> снятия заявки           |

| Параметр           | Тип    | Описание  |
|--------------------|--------|---|
| bank_acc_id        | STRING | Идентификатор расчетного счета/кода в клиринговой организации   |
| value_entry_type   | NUMBER | Способ указания объема заявки. Возможные значения: <ul style="list-style-type: none"> <li>«0» – по количеству,</li> <li>«1» – по объему</li> </ul>  |
| repoterm           | NUMBER | Срок РЕПО, в календарных днях   |
| repovalue          | NUMBER | Сумма РЕПО на текущую дату. Отображается с точностью 2 знака  |
| repo2value         | NUMBER | Объем сделки выкупа РЕПО. Отображается с точностью 2 знака  |
| repo_value_balance | NUMBER | Остаток суммы РЕПО за вычетом суммы привлеченных или предоставленных по сделке РЕПО денежных средств в неисполненной части заявки, по состоянию на текущую дату. Отображается с точностью 2 знака   |
| start_discount     | NUMBER | Начальный дисконт, в %  |
| reject_reason      | STRING | Причина отклонения заявки брокером  |
| ext_order_flags    | NUMBER | Битовое поле для получения специфических параметров с западных площадок   |
| min_qty            | NUMBER | Минимально допустимое количество, которое можно указать в заявке по данному инструменту. Если имеет значение 0, значит ограничение по количеству не задано  |
| exec_type          | NUMBER | Тип исполнения заявки. Возможные значения: <ul style="list-style-type: none"> <li>«0» – «Значение не указано»;</li> <li>«1» – «Немедленно или отклонить»;</li> <li>«2» – «Поставить в очередь»;</li> <li>«3» – «Снять остаток»;</li> <li>«4» – «До снятия»;</li> <li>«5» – «До даты»;</li> <li>«6» – «В течение сессии»;</li> <li>«7» – «Открытие»;</li> <li>«8» – «Закрытие»;</li> <li>«9» – «Кросс»;</li> <li>«11» – «До следующей сессии»;</li> <li>«13» – «До отключения»;</li> <li>«15» – «До времени»;</li> <li>«16» – «Следующий аукцион»</li> </ul> |
| side_qualifier     | NUMBER | Поле для получения параметров по западным площадкам. Если имеет значение «0», значит значение не задано   |
| acnt_type          | NUMBER | Поле для получения параметров по западным площадкам. Если имеет значение «0», значит значение не задано   |

| Параметр           | Тип    | Описание   |
|--------------------|--------|--|
| capacity           | NUMBER | Поле для получения параметров по западным площадкам.<br>Если имеет значение «0», значит значение не задано |
| passive_only_order | NUMBER | Поле для получения параметров по западным площадкам.<br>Если имеет значение «0», значит значение не задано |

## 4.6 Текущие позиции по клиентским счетам

Описание параметров Таблицы текущих позиций по счетам:

| Параметр       | Тип    | Описание   |
|----------------|--------|--|
| firmid         | STRING | Идентификатор фирмы  |
| sec_code       | STRING | Код бумаги   |
| trdaccid       | STRING | Торговый счет  |
| depaccid       | STRING | Счет депо  |
| openbal        | NUMBER | Входящий остаток   |
| currentpos     | NUMBER | Текущий остаток  |
| plannedpossell | NUMBER | Плановая продажа   |
| plannedposbuy  | NUMBER | Плановая покупка   |
| planbal        | NUMBER | Контрольный остаток простого клиринга, равен входящему остатку минус плановая позиция на продажу, включенная в простой клиринг |
| usqtyb         | NUMBER | Куплено  |
| usqtys         | NUMBER | Продано  |
| planned        | NUMBER | Плановый остаток, равен текущему остатку минус плановая позиция на продажу   |
| settlebal      | NUMBER | Плановая позиция после проведения расчетов   |
| bank_acc_id    | STRING | Идентификатор расчетного счета/кода в клиринговой организации  |
| firmuse        | NUMBER | Признак счета обеспечения. Возможные значения:<br>_ «0» – для обычных счетов,<br>_ «1» – для счета обеспечения                 |

## 4.7 Текущие позиции по бумагам

Описание параметров Таблицы текущих позиций по бумагам:

| Параметр       | Тип    | Описание   |
|----------------|--------|--|
| firmid         | STRING | Фирма  |
| seccode        | STRING | Код бумаги   |
| openbal        | NUMBER | Входящий остаток                                   |
| currentpos     | NUMBER | Текущий остаток                                    |
| plannedposbuy  | NUMBER | Объем активных заявок на покупку, в ценных бумагах |
| plannedpossell | NUMBER | Объем активных заявок на продажу, в ценных бумагах |
| usqtyb         | NUMBER | Куплено  |
| usqtys         | NUMBER | Продано  |

## 4.8 Стоп-заявки

Описание параметров Таблицы стоп-заявок:

| Параметр        | Тип    | Описание   |
|-----------------|--------|--|
| order_num       | NUMBER | Регистрационный номер стоп-заявки на сервере QUIK                                    |
| ordertime       | NUMBER | Время выставления  |
| flags           | NUMBER | <a href="#">Набор битовых флагов</a>   |
| brokerref       | STRING | Комментарий, обычно: <код клиента>/<номер поручения>                                 |
| firmid          | STRING | Идентификатор дилер  |
| account         | STRING | Торговый счет  |
| condition       | NUMBER | Направленность стоп-цены. Возможные значения:<br>_ «4» – «<=»,<br>_ «5» – «>=»       |
| condition_price | NUMBER | Стоп-цена  |
| price           | NUMBER | Цена   |
| qty             | NUMBER | Количество в лотах   |
| linkedorder     | NUMBER | Номер заявки в торговой системе, зарегистрированной по наступлению условия стоп-цены |

| Параметр             | Тип    | Описание   |
|----------------------|--------|--|
| expiry               | NUMBER | Дата окончания срока действия заявки   |
| trans_id             | NUMBER | Идентификатор транзакции   |
| client_code          | STRING | Код клиента  |
| co_order_num         | NUMBER | Связанная заявка   |
| co_order_price       | NUMBER | Цена связанной заявки  |
| stop_order_type      | NUMBER | Вид стоп заявки. Возможные значения: <ul style="list-style-type: none"> <li>«1» – стоп-лимит,</li> <li>«2» – условие по другому инструменту,</li> <li>«3» – со связанной заявкой,</li> <li>«6» – тейк-профит,</li> <li>«7» – стоп-лимит по исполнению активной заявки,</li> <li>«8» – тейк-профит по исполнению активной заявки</li> <li>«9» – тейк-профит и стоп-лимит</li> </ul> |
| orderdate            | NUMBER | Дата выставления   |
| alltrade_num         | NUMBER | Сделка условия   |
| stopflags            | NUMBER | <a href="#">Набор битовых флагов</a>   |
| offset               | NUMBER | Отступ от min/max  |
| spread               | NUMBER | Защитный спред   |
| balance              | NUMBER | Активное количество  |
| uid                  | NUMBER | Идентификатор пользователя   |
| filled_qty           | NUMBER | Исполненное количество   |
| withdraw_time        | NUMBER | Время снятия заявки  |
| condition_price2     | NUMBER | Стоп-лимит цена (для заявок типа «Тэйк-профит и стоп-лимит»)   |
| active_from_time     | NUMBER | Время начала периода действия заявки типа «Тэйк-профит и стоп-лимит»   |
| active_to_time       | NUMBER | Время окончания периода действия заявки типа «Тэйк-профит и стоп-лимит»  |
| sec_code             | STRING | Код бумаги заявки  |
| class_code           | STRING | Код класса заявки  |
| condition_sec_code   | STRING | Код бумаги стоп-цены   |
| condition_class_code | STRING | Код класса стоп-цены   |

| Параметр          | Тип    | Описание   |
|-------------------|--------|--|
| canceled_uid      | NUMBER | Идентификатор пользователя, снявшего стоп-заявку |
| order_date_time   | TABLE  | Время выставления стоп-заявки                    |
| withdraw_datetime | TABLE  | Время снятия стоп-заявки                         |

## 4.9 Лимиты по фьючерсам

Описание параметров фьючерсного лимита:

| Параметр               | Тип    | Описание  |
|------------------------|--------|---|
| firmid                 | STRING | Идентификатор фирмы   |
| trdaccid               | STRING | Торговый счет   |
| limit_type             | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"> <li>«0» – «Денежные средства»;</li> <li>«1» – «Залоговые денежные средства»;</li> <li>«2» – «По совокупным средствам»;</li> <li>«3» – «Клиринговые денежные средства»;</li> <li>«4» – «Клиринговые залоговые денежные средства»;</li> <li>«5» – «Лимит открытых позиций на спот-рынке»;</li> <li>«6» – «Суммарные залоговые средства в иностранной валюте (в рублях)»;</li> <li>«7» – «Залоговые средства в иностранной валюте»</li> </ul> |
| liquidity_coef         | NUMBER | Коэффициент ликвидности   |
| cbp_prev_limit         | NUMBER | Предыдущий лимит открытых позиций   |
| cbplimit               | NUMBER | Лимит открытых позиций  |
| cbplused               | NUMBER | Текущие чистые позиции  |
| cbplplanned            | NUMBER | Плановые чистые позиции   |
| varmargin              | NUMBER | Вариационная маржа  |
| accruedint             | NUMBER | Накопленный купонный доход  |
| cbplused_for_orders    | NUMBER | Текущие чистые позиции (под заявки)   |
| cbplused_for_positions | NUMBER | Текущие чистые позиции (под открытые позиции)   |
| options_premium        | NUMBER | Премия по опционам  |
| ts_comission           | NUMBER | Биржевые сборы  |
| kgo                    | NUMBER | Коэффициент клиентского гарантийного обеспечения  |

| Параметр       | Тип    | Описание  |
|----------------|--------|---|
| currcode       | STRING | Валюта, в которой транслируется ограничение   |
| real_varmargin | NUMBER | Реально начисленная в ходе клиринга вариационная маржа. Отображается с точностью до 2 двух знаков. При этом, поле «varmargin» транслируется вариационная маржа, рассчитанная с учетом установленных границ изменения цены |

## 4.10 Позиции по клиентским счетам (фьючерсы)

Описание параметров Таблицы позиций по фьючерсам:

| Параметр      | Тип    | Описание   |
|---------------|--------|--|
| firmid        | STRING | Идентификатор фирмы  |
| trdaccid      | STRING | Торговый счет  |
| sec_code      | STRING | Код фьючерсного контракта  |
| type          | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"> <li>— «Основной счет»;</li> <li>— «Клиентские и дополнительные счета»;</li> <li>— «Все счета торг. членов»;</li> <li>— &lt;пусто&gt;</li> </ul> |
| startbuy      | NUMBER | Входящие длинные позиции   |
| startsell     | NUMBER | Входящие короткие позиции  |
| startnet      | NUMBER | Входящие чистые позиции  |
| todaybuy      | NUMBER | Текущие длинные позиции  |
| todaysell     | NUMBER | Текущие короткие позиции   |
| totalnet      | NUMBER | Текущие чистые позиции   |
| openbuys      | NUMBER | Активные на покупку  |
| opensells     | NUMBER | Активные на продажу  |
| cbplused      | NUMBER | Оценка текущих чистых позиций  |
| cbplplanned   | NUMBER | Плановые чистые позиции  |
| varmargin     | NUMBER | Вариационная маржа   |
| avrposnprice  | NUMBER | Эффективная цена позиций   |
| positionvalue | NUMBER | Стоимость позиций  |



| Параметр        | Тип    | Описание  |
|-----------------|--------|---|
| real_varmargin  | NUMBER | Реально начисленная в ходе клиринга вариационная маржа. Отображается с точностью до 2 двух знаков. При этом в уже имеющемся поле «varmargin» транслируется вариационная маржа, рассчитанная с учетом установленных границ изменения цены  |
| total_varmargin | NUMBER | Суммарная вариационная маржа по итогам основного клиринга начисленная по всем позициям. Отображается с точностью до 2 двух знаков   |
| session_status  | NUMBER | Актуальный статус торговой сессии. Возможные значения:<br>«0» – основная / вечерняя сессия;<br>«1» – начался промклиринг;<br>«2» – завершился промклиринг;<br>«3» – начался основной клиринг;<br>«4» – основной клиринг, новая сессия назначена;<br>«5» – завершился основной клиринг |

## 4.11 Лимиты по денежным средствам

Описание параметров Таблицы лимитов по денежным средствам:

| Параметр            | Тип    | Описание  |
|---------------------|--------|---|
| currcode            | STRING | Код валюты  |
| tag                 | STRING | Тег расчетов  |
| firmid              | STRING | Идентификатор фирмы   |
| client_code         | STRING | Код клиента   |
| openbal             | NUMBER | Входящий остаток по деньгам                                 |
| openlimit           | NUMBER | Входящий лимит по деньгам                                   |
| currentbal          | NUMBER | Текущий остаток по деньгам                                  |
| currentlimit        | NUMBER | Текущий лимит по деньгам                                    |
| locked              | NUMBER | Заблокированное количество                                  |
| locked_value_coef   | NUMBER | Стоимость активов в заявках на покупку немаржинальных бумаг |
| locked_margin_value | NUMBER | Стоимость активов в заявках на покупку маржинальных бумаг   |
| leverage            | NUMBER | Плечо   |

| Параметр   | Тип    | Описание  |
|------------|--------|---|
| limit_kind | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"> <li>— «0» – обычные лимиты,</li> <li>— иначе – технологические лимиты</li> </ul> |

## 4.12 Удаление денежного лимита

Описание параметров таблицы Удаление денежного лимита:

| Параметр    | Тип    | Описание  |
|-------------|--------|---|
| currcode    | STRING | Код валюты  |
| tag         | STRING | Тег расчетов  |
| client_code | STRING | Код клиента   |
| firmid      | STRING | Идентификатор фирмы   |
| limit_kind  | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"> <li>— «0» – обычные лимиты,</li> <li>— иначе – технологические лимиты</li> </ul> |

## 4.13 Удаление бумажного лимита

Описание параметров таблицы Удаление бумажного лимита:

| Параметр    | Тип    | Описание   |
|-------------|--------|--|
| sec_code    | STRING | Код инструмента  |
| trdaccid    | STRING | Код торгового счета  |
| firmid      | STRING | Идентификатор фирмы  |
| client_code | STRING | Код клиента  |
| limit_kind  | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"> <li>— «0» – обычные лимиты,</li> <li>— значение не равное «0» – технологические лимиты</li> </ul> |

## 4.14 Удаление фьючерсного лимита

Описание параметров таблицы Удаление фьючерсного лимита:

| Параметр   | Тип    | Описание            |
|------------|--------|---------------------|
| firmid     | STRING | Идентификатор фирмы |
| limit_type | STRING | Тип лимита          |

## 4.15 Лимиты по бумагам

Описание параметров Таблицы лимитов по бумагам:

| Параметр           | Тип    | Описание  |
|--------------------|--------|---|
| sec_code           | STRING | Код бумаги  |
| trdaccid           | STRING | Счет депо   |
| firmid             | STRING | Идентификатор фирмы   |
| client_code        | STRING | Код клиента   |
| openbal            | NUMBER | Входящий остаток по бумагам   |
| openlimit          | NUMBER | Входящий лимит по бумагам   |
| currentbal         | NUMBER | Текущий остаток по бумагам  |
| currentlimit       | NUMBER | Текущий лимит по бумагам  |
| locked_sell        | NUMBER | Заблокировано на продажу количества лотов   |
| locked_buy         | NUMBER | Заблокировано на покупку количества лотов   |
| locked_buy_value   | NUMBER | Стоимость ценных бумаг, заблокированных под покупку   |
| locked_sell_value  | NUMBER | Стоимость ценных бумаг, заблокированных под продажу   |
| awg_position_price | NUMBER | Цена приобретения   |
| limit_kind         | NUMBER | Тип лимита. Возможные значения: <ul style="list-style-type: none"><li>— «0» – обычные лимиты,</li><li>— значение не равное «0» – технологические лимиты</li></ul> |

## 4.16 Денежные позиции

Описание параметров Таблицы денежных позиций:

| Параметр    | Тип    | Описание  |
|-------------|--------|---|
| firmid      | STRING | Идентификатор фирмы                             |
| currcode    | STRING | Код валюты                                      |
| tag         | STRING | Тег расчетов                                    |
| description | STRING | Описание  |
| openbal     | NUMBER | Входящий остаток                                |
| currentpos  | NUMBER | Текущий остаток                                 |
| plannedpos  | NUMBER | Плановый остаток                                |
| limit1      | NUMBER | Внешнее ограничение по деньгам                  |
| limit2      | NUMBER | Внутреннее (собственное) ограничение по деньгам |
| orderbuy    | NUMBER | В заявках на продажу                            |
| ordersell   | NUMBER | В заявках на покупку                            |
| netto       | NUMBER | Нетто-позиция                                   |
| plannedbal  | NUMBER | Плановая позиция                                |
| debit       | NUMBER | Дебит   |
| credit      | NUMBER | Кредит  |
| bank_acc_id | STRING | Идентификатор счета                             |
| margincall  | NUMBER | Маржинальное требование на начало торгов        |
| settlebal   | NUMBER | Плановая позиция после проведения расчетов      |

## 4.17 Заявки на внебиржевые сделки

Описание параметров Таблицы заявок на внебиржевые сделки:

| Параметр      | Тип    | Описание                             |
|---------------|--------|--------------------------------------|
| neg_deal_num  | NUMBER | Номер                                |
| neg_deal_time | NUMBER | Время выставления заявки             |
| flags         | NUMBER | <a href="#">Набор битовых флагов</a> |

| Параметр       | Тип    | Описание   |
|----------------|--------|--|
| brokerref      | STRING | Комментарий, обычно: <код клиента>/<номер поручения>   |
| userid         | STRING | Треjder  |
| firmid         | STRING | Идентификатор дилера   |
| cpuserid       | STRING | Треjder партнера   |
| cpfirmid       | STRING | Код фирмы партнера   |
| account        | STRING | Счет   |
| price          | NUMBER | Цена   |
| qty            | NUMBER | Количество   |
| matchref       | STRING | Ссылка   |
| settlecode     | STRING | Код расчетов   |
| yield          | NUMBER | Доходность   |
| accruedint     | NUMBER | Купонный процент   |
| value          | NUMBER | Объем  |
| price2         | NUMBER | Цена выкупа  |
| reporate       | NUMBER | Ставка РЕПО (%)  |
| refundrate     | NUMBER | Ставка возмещения (%)  |
| trans_id       | NUMBER | ID транзакции  |
| client_code    | STRING | Код клиента  |
| repoentry      | NUMBER | Тип ввода заявки РЕПО. Возможные значения: <ul style="list-style-type: none"> <li>– «0» – «Не определен»;</li> <li>– «1» – «Цена1+Ставка»;</li> <li>– «2» – «Ставка+ Цена2»;</li> <li>– «3» – «Цена1+Цена2»;</li> <li>– «4» – «Сумма РЕПО + Количество»;</li> <li>– «5» – «Сумма РЕПО + Дисконт»;</li> <li>– «6» – «Количество + Дисконт»;</li> <li>– «7» – «Сумма РЕПО»;</li> <li>– «8» – «Количество»</li> </ul> |
| repovalue      | NUMBER | Сумма РЕПО   |
| repo2value     | NUMBER | Объем выкупа РЕПО  |
| repoterm       | NUMBER | Срок РЕПО  |
| start_discount | NUMBER | Начальный дисконт (%)  |

| Параметр                 | Тип    | Описание  |
|--------------------------|--------|---|
| lower_discount           | NUMBER | Нижний дисконт (%)  |
| upper_discount           | NUMBER | Верхний дисконт (%)   |
| block_securities         | NUMBER | Блокировка обеспечения («Да»/«Нет»)                           |
| uid                      | NUMBER | Идентификатор пользователя                                    |
| withdraw_time            | NUMBER | Время снятия заявки   |
| neg_deal_date            | NUMBER | Дата выставления заявки                                       |
| balance                  | NUMBER | Остаток   |
| origin_repoalue          | NUMBER | Сумма РЕПО первоначальная                                     |
| origin_qty               | NUMBER | Количество первоначальное                                     |
| origin_discount          | NUMBER | Процент дисконта первоначальный                               |
| neg_deal_activation_date | NUMBER | Дата активации заявки   |
| neg_deal_activation_time | NUMBER | Время активации заявки  |
| quoteno                  | NUMBER | Встречная безадресная заявка                                  |
| settle_currency          | NUMBER | Валюта расчетов   |
| sec_code                 | STRING | Код бумаги  |
| class_code               | STRING | Код класса  |
| bank_acc_id              | STRING | Идентификатор расчетного счета/кода в клиринговой организации |
| withdraw_date            | NUMBER | Дата снятия адресной заявки в формате «ГГГГММДД»              |
| linkedorder              | NUMBER | Номер предыдущей заявки. Отображается с точностью «0»         |
| activation_date_time     | TABLE  | Дата и время активации заявки                                 |
| withdraw_date_time       | TABLE  | Дата и время снятия заявки                                    |
| date_time                | TABLE  | Дата и время заявки   |

## 4.18 Сделки для исполнения

Описание параметров Таблицы сделок для исполнения:

| Параметр     | Тип    | Описание   |
|--------------|--------|--|
| trade_num    | NUMBER | Номер сделки   |
| trade_date   | NUMBER | Дата торгов  |
| settle_date  | NUMBER | Дата расчетов  |
| flags        | NUMBER | <a href="#">Набор битовых флагов</a>                 |
| brokerref    | STRING | Комментарий, обычно: <код клиента>/<номер поручения> |
| firmid       | STRING | Идентификатор дилера                                 |
| account      | STRING | Счет депо  |
| cpfirmid     | STRING | Код фирмы партнера                                   |
| cpaccount    | STRING | Счет депо партнера                                   |
| price        | NUMBER | Цена   |
| qty          | NUMBER | Количество   |
| value        | NUMBER | Объем  |
| settlecode   | STRING | Код расчетов   |
| report_num   | NUMBER | Отчет  |
| cpreport_num | NUMBER | Отчет партнера                                       |
| accruedint   | NUMBER | Купонный процент                                     |
| repotradeno  | NUMBER | Номер сделки 1-ой части РЕПО                         |
| price1       | NUMBER | Цена 1-ой части РЕПО                                 |
| reporate     | NUMBER | Ставка РЕПО (%)                                      |
| price2       | NUMBER | Цена выкупа  |
| client_code  | STRING | Код клиента  |
| ts_comission | NUMBER | Комиссия торговой системы                            |
| balance      | NUMBER | Остаток  |
| settle_time  | NUMBER | Время исполнения                                     |
| amount       | NUMBER | Сумма обязательства                                  |

| Параметр              | Тип    | Описание   |
|-----------------------|--------|--|
| repovalue             | NUMBER | Сумма РЕПО   |
| repoterm              | NUMBER | Срок РЕПО  |
| repo2value            | NUMBER | Объем выкупа РЕПО  |
| return_value          | NUMBER | Сумма возврата РЕПО  |
| discount              | NUMBER | Дисконт (%)  |
| lower_discount        | NUMBER | Нижний дисконт (%)   |
| upper_discount        | NUMBER | Верхний дисконт (%)  |
| block_securities      | NUMBER | Блокировать обеспечение («Да»/«Нет»)   |
| urgency_flag          | NUMBER | Исполнить («Да»/«Нет»)   |
| type                  | NUMBER | Тип. Возможные значения: <ul style="list-style-type: none"> <li>– «0» – «Внесистемная сделка»,</li> <li>– «1» – «Первая часть сделки РЕПО»,</li> <li>– «2» – «Вторая часть сделки РЕПО»,</li> <li>– «3» – «Компенсационный взнос»,</li> <li>– «4» – «Дефолтер: отложенные обязательства и требования»,</li> <li>– «5» – «Пострадавший: отложенные обязательства и требования»</li> </ul> |
| operation_type        | NUMBER | Направленность. Возможные значения: <ul style="list-style-type: none"> <li>– «1» – «Зачислить»,</li> <li>– «2» – «Списать»</li> </ul>  |
| expected_discount     | NUMBER | Дисконт после взноса (%)   |
| expected_quantity     | NUMBER | Количество после взноса  |
| expected_repovalue    | NUMBER | Сумма РЕПО после взноса  |
| expected_repo2value   | NUMBER | Стоимость выкупа после взноса  |
| expected_return_value | NUMBER | Сумма возврата после взноса  |
| order_num             | NUMBER | Номер заявки   |
| report_trade_date     | NUMBER | Дата заключения  |
| settled               | NUMBER | Состояние расчетов по сделке. Возможные значения: <ul style="list-style-type: none"> <li>– «1» – «Processed»,</li> <li>– «2» – «Not processed»,</li> <li>– «3» – «Is processing»</li> </ul>  |
| clearing_type         | NUMBER | Тип клиринга. Возможные значения: <ul style="list-style-type: none"> <li>– «1» – «Not set»,</li> <li>– «2» – «Simple»,</li> <li>– «3» – «Multilateral»</li> </ul>  |



| Параметр               | Тип    | Описание   |
|------------------------|--------|--|
| report_comission       | NUMBER | Комиссия за отчет  |
| coupon_payment         | NUMBER | Купонная выплата   |
| principal_payment      | NUMBER | Выплата по основному долгу   |
| principal_payment_date | NUMBER | Дата выплаты по основному долгу  |
| nextdaysettle          | NUMBER | Дата следующего дня расчетов   |
| settle_currency        | STRING | Валюта расчетов  |
| sec_code               | STRING | Код бумаги   |
| class_code             | STRING | Код класса   |
| compval                | NUMBER | Сумма отступного в валюте сделки   |
| parenttradeno          | NUMBER | Идентификационный номер витринной сделки   |
| bankid                 | STRING | Расчетная организация  |
| bankaccid              | STRING | Код позиции  |
| precisebalance         | NUMBER | Количество бумаг к исполнению (в лотах)  |
| confirmtime            | NUMBER | Время подтверждения в формате «ЧЧММСС»   |
| ex_flags               | NUMBER | Расширенные флаги сделки для исполнения. Возможные значения:<br><div> <div>– «1» – «Подтверждена контрагентом»;</div> <div>– «2» – «Подтверждена»</div> </div> |
| confirmreport          | NUMBER | Номер поручения  |

## 4.19 Торговые счета

Описание параметров таблицы Торговые счета:

| Параметр        | Тип    | Описание  |
|-----------------|--------|---|
| class_codes     | STRING | Список кодов классов, разделенных символом « »  |
| firmid          | STRING | Идентификатор фирмы   |
| trdaccid        | STRING | Код торгового счета   |
| description     | STRING | Описание  |
| fullcoveredsell | NUMBER | Запрет необеспеченных продаж. Возможные значения:<br><div> <div>– «0» – Нет;</div> <div>– «1» – Да</div> </div> |

| Параметр      | Тип    | Описание  |
|---------------|--------|---|
| main_trdaccid | NUMBER | Номер основного торгового счета   |
| bankid_t0     | STRING | Расчетная организация по «Т0»   |
| bankid_tplus  | STRING | Расчетная организация по «Т+»   |
| trdacc_type   | NUMBER | Тип торгового счета   |
| depunitid     | STRING | Раздел счета Депо   |
| status        | NUMBER | Статус торгового счета. Возможные значения: <ul style="list-style-type: none"> <li>«0» – операции разрешены;</li> <li>«1» – операции запрещены</li> </ul> |
| firmuse       | NUMER  | Тип раздела. Возможные значения: <ul style="list-style-type: none"> <li>«0» – раздел обеспечения;</li> <li>иначе – для торговых разделов</li> </ul>       |
| depaccid      | STRING | Номер счета депо в депозитарии  |
| bank_acc_id   | STRING | Код дополнительной позиции по денежным средствам  |

## 4.20 Отчеты по сделкам для исполнения

Описание параметров таблицы Отчеты по сделкам для исполнения:

| Параметр      | Тип    | Описание                             |
|---------------|--------|--------------------------------------|
| report_num    | NUMBER | Отчет                                |
| report_date   | NUMBER | Дата отчета                          |
| flags         | NUMBER | <a href="#">Набор битовых флагов</a> |
| userid        | STRING | Идентификатор пользователя           |
| firmid        | STRING | Идентификатор фирмы                  |
| account       | STRING | Счет депо                            |
| cpfirmid      | STRING | Код фирмы партнера                   |
| craccount     | STRING | Код торгового счета партнера         |
| qty           | NUMBER | Количество бумаг, в лотах            |
| value         | NUMBER | Объем сделки, выраженный в рублях    |
| withdraw_time | NUMBER | Время снятия заявки                  |
| report_type   | NUMBER | Тип отчета                           |

| Параметр         | Тип    | Описание                                   |
|------------------|--------|--|
| report_kind      | NUMBER | Вид отчета                                 |
| commission       | NUMBER | Объем комиссии по сделке, выраженный в руб |
| sec_code         | STRING | Код бумаги                                 |
| class_code       | STRING | Код класса                                 |
| report_time      | NUMBER | Время отчета                               |
| report_date_time | TABLE  | Дата и время отчета                        |

## 4.21 Инструменты

Описание параметров таблицы Инструменты:

| Параметр       | Тип    | Описание  |
|----------------|--------|---|
| code           | STRING | Код инструмента                                   |
| name           | STRING | Наименование инструмента                          |
| short_name     | STRING | Короткое наименование инструмента                 |
| class_code     | STRING | Код класса инструментов                           |
| class_name     | STRING | Наименование класса инструментов                  |
| face_value     | NUMBER | Номинал   |
| face_unit      | STRING | Валюта номинала                                   |
| scale          | NUMBER | Точность (количество значащих цифр после запятой) |
| mat_date       | NUMBER | Дата погашения                                    |
| lot_size       | NUMBER | Размер лота                                       |
| isin_code      | STRING | ISIN  |
| min_price_step | NUMBER | Минимальный шаг цены                              |

## 4.22 Свечки графика

Описание параметров свечки графика:

| Параметр  | Тип    | Описание   |
|-----------|--------|--|
| open      | NUMBER | Цена открытия  |
| close     | NUMBER | Цена закрытия  |
| high      | NUMBER | Максимальная цена сделки   |
| low       | NUMBER | Минимальная цена сделки  |
| volume    | NUMBER | Объем последней сделки   |
| datetime  | TABLE  | <a href="#">Формат даты и времени</a>  |
| doesExist | NUMBER | Признак расчета индикатора при наличии свечки. Возможные значения: <ul style="list-style-type: none"><li>— «0» – индикатор не рассчитан,</li><li>— «1» – индикатор рассчитан</li></ul> |

## 4.23 Формат даты и времени, используемый в таблицах

Описание формата даты и времени, используемого в некоторых таблицах:

| Параметр | Тип    | Описание         |
|----------|--------|------------------|
| mcs      | NUMBER | Микросекунды     |
| ms       | NUMBER | Миллисекунды     |
| sec      | NUMBER | Секунды          |
| min      | NUMBER | Минуты           |
| hour     | NUMBER | Часы             |
| day      | NUMBER | День             |
| week_day | NUMBER | Номер дня недели |
| month    | NUMBER | Месяц            |
| year     | NUMBER | Год              |

Для корректного отображения даты и времени, эти параметры должны быть заданы.

## 4.24 Транзакции

Описание параметров транзакций.

| Параметр        | Тип    | Описание  |
|-----------------|--------|---|
| trans_id        | NUMBER | Пользовательский идентификатор транзакции   |
| status          | NUMBER | Статус транзакции. Возможные значения: <ul style="list-style-type: none"><li>«0» – транзакция отправлена серверу,</li><li>«1» – транзакция получена на сервер QUIK от клиента,</li><li>«2» – ошибка при передаче транзакции в торговую систему. Так как отсутствует подключение шлюза Московской Биржи, повторно транзакция не отправляется,</li><li>«3» – транзакция выполнена,</li><li>«4» – транзакция не выполнена торговой системой. Более подробное описание ошибки отражается в поле «Сообщение»,</li><li>«5» – транзакция не прошла проверку сервера QUIK по каким-либо критериям. Например, проверку на наличие прав у пользователя на отправку транзакции данного типа,</li><li>«6» – транзакция не прошла проверку лимитов сервера QUIK,</li><li>«10» – транзакция не поддерживается торговой системой,</li><li>«11» – транзакция не прошла проверку правильности электронной цифровой подписи,</li><li>«12» – не удалось дождаться ответа на транзакцию, т.к. истек таймаут ожидания. Может возникнуть при подаче транзакций из QPILE,</li><li>«13» – транзакция отвергнута, так как ее выполнение могло привести к кросс-сделке (т.е. сделке с тем же самым клиентским счетом)</li></ul> |
| result_msg      | STRING | Сообщение   |
| time            | NUMBER | Время   |
| uid             | NUMBER | Идентификатор   |
| flags           | NUMBER | Флаги транзакции (временно не используется)   |
| server_trans_id | NUMBER | Идентификатор транзакции на сервере   |
| *order_num      | NUMBER | Номер заявки  |
| *price          | NUMBER | Цена  |
| *quantity       | NUMBER | Количество  |

| Параметр       | Тип    | Описание              |
|----------------|--------|-----------------------|
| *balance       | NUMBER | Остаток               |
| *firm_id       | STRING | Идентификатор фирмы   |
| *account       | STRING | Торговый счет         |
| *client_code   | STRING | Код клиента           |
| *brokerref     | STRING | Поручение             |
| *class_code    | STRING | Код класса            |
| *sec_code      | STRING | Код бумаги            |
| *exchange_code | STRING | Биржевой номер заявки |

\* – параметр может иметь значение nil

## 5. Описание битовых флагов

### 5.1 Флаги для таблиц Заявки, Заявки на внебиржевые сделки, Сделки, Сделки для исполнения

| Флаг установлен | Значение   |
|-----------------|--|
| бит 0 (0x1)     | Заявка активна, иначе – не активна   |
| бит 1 (0x2)     | Заявка снята. Если флаг не установлен и значение бита «0» равно «0», то заявка исполнена                                       |
| бит 2 (0x4)     | Заявка на продажу, иначе – на покупку. Данный флаг для сделок и сделок для исполнения определяет направление сделки (BUY/SELL) |
| бит 3 (0x8)     | Заявка лимитированная, иначе – рыночная  |
| бит 4 (0x10)    | Разрешить / запретить сделки по разным ценам   |
| бит 5 (0x20)    | Исполнить заявку немедленно или снять (FILL OR KILL)   |
| бит 6 (0x40)    | Заявка маркет-мейкера. Для адресных заявок – заявка отправлена контрагенту   |
| бит 7 (0x80)    | Для адресных заявок – заявка получена от контрагента   |
| бит 8 (0x100)   | Снять остаток  |

| Флаг установлен | Значение       |
|-----------------|----------------|
| бит 9 (0x200)   | Айсберг-заявка |

## 5.2 Флаги для таблицы Обезличенные сделки

| Флаг установлен | Значение          |
|-----------------|-------------------|
| бит 0 (0x1)     | Сделка на продажу |
| бит 1 (0x2)     | Сделка на покупку |

Если флаги не установлены, направление сделки не определено.

## 5.3 Флаги для таблицы Стоп-заявки

| Флаг установлен | Значение   |
|-----------------|--|
| бит 0 (0x1)     | Заявка активна, иначе не активна   |
| бит 1 (0x2)     | Заявка снята. Если не установлен и значение бита 0 равно 0, то заявка исполнена  |
| бит 2 (0x4)     | Заявка на продажу, иначе – на покупку  |
| бит 3 (0x8)     | Лимитированная заявка  |
| бит 5 (0x20)    | Стоп-заявка ожидает активации  |
| бит 6 (0x40)    | Стоп-заявка с другого сервера  |
| бит 8 (0x100)   | Устанавливается в случае стоп-заявки типа тейк-профита по заявке, в случае когда исходная заявка частично исполнена и по выставленной тейк-профит заявке на исполненную часть заявки выполнилось условие активации |
| бит 9 (0x200)   | Стоп-заявка активирована вручную   |
| бит 10 (0x400)  | Стоп-заявка сработала, но была отвергнута торговой системой  |
| бит 11 (0x800)  | Стоп-заявка сработала, но не прошла контроль лимитов   |
| бит 12 (0x1000) | Стоп-заявка снята, так как снята связанная заявка  |
| бит 13 (0x2000) | Стоп-заявка снята, так как связанная заявка исполнена  |
| бит 15 (0x8000) | Идет расчет минимума-максимума   |

## 5.4 Дополнительные флаги для таблицы Стоп-заявки

| Флаг установлен | Значение   |
|-----------------|--|
| бит 0 (0x1)     | Использовать остаток основной заявки                               |
| бит 1 (0x2)     | При частичном исполнении заявки снять стоп-заявку                  |
| бит 2 (0x4)     | Активировать стоп-заявку при частичном исполнении связанной заявки |
| бит 3 (0x8)     | Отступ задан в процентах, иначе – в пунктах цены                   |
| бит 4 (0x10)    | Защитный спред задан в процентах, иначе – в пунктах цены           |
| бит 5 (0x20)    | Срок действия стоп-заявки ограничен сегодняшним днем               |
| бит 6 (0x40)    | Установлен интервал времени действия стоп-заявки                   |
| бит 7 (0x80)    | Выполнение тейк-профита по рыночной цене                           |
| бит 8 (0x100)   | Выполнение стоп-заявки по рыночной цене                            |

## 5.5 Обязательства и требования по активам

Описание параметров Таблицы обязательств и требований по активам:

| Параметр     | Тип    | Описание  |
|--------------|--------|---|
| firmid       | STRING | Идентификатор фирмы   |
| depo_account | STRING | Номер счета депо в Депозитарии (НДЦ)                          |
| account      | STRING | Торговый счет   |
| bank_acc_id  | STRING | Идентификатор расчетного счета/кода в клиринговой организации |
| settle_date  | NUMBER | Дата расчетов   |
| qty          | NUMBER | Количество ценных бумаг в сделках                             |
| qty_buy      | NUMBER | Количество ценных бумаг в заявках на покупку                  |
| qty_sell     | NUMBER | Количество ценных бумаг в заявках на продажу                  |
| netto        | NUMBER | Нетто-позиция   |
| debit        | NUMBER | Дебит   |
| credit       | NUMBER | Кредит  |



| Параметр   | Тип    | Описание          |
|------------|--------|-------------------|
| sec_code   | STRING | Код бумаги заявки |
| class_code | STRING | Код класса заявки |

## 6. Функции для работы с битовыми масками в структурах данных

### 6.1 bit.tohex

Функция конвертирует первый аргумент в шестнадцатеричную строку. Количество знаков в строке задается вторым необязательным параметром.

Формат вызова:

```
STRING bit.tohex(NUMBERx [, NUMBER n])
```

### 6.2 bit.bnot

Функция возвращает результат битовой операции NOT над аргументом x.

Формат вызова:

```
NUMBER bit.bnot(NUMBER x)
```

### 6.3 bit.band

Функция возвращает результат битовой операции AND над аргументами. Аргументов может быть несколько, при этом обязательные аргументы **x1** и **x2**.

Формат вызова:

```
NUMBER bit.band(NUMBER x1, NUMBER x2, ...)
```

### 6.4 bit.bor

Функция возвращает результат битовой операции OR (ИЛИ) над аргументами. Аргументов может быть несколько, при этом обязательные аргументы **x1** и **x2**.

Формат вызова:

```
NUMBER bit.bor(NUMBER x1, NUMBER x2,...)
```

## 6.5 bit.bxor

Функция возвращает результат битовой операции XOR (исключающее ИЛИ) над аргументами. Аргументов может быть несколько, при этом обязательные аргументы **x1** и **x2**.

Формат вызова:

```
NUMBER bit.bxor(NUMBER x1, NUMBER x2,...)
```

# 7. Индикаторы технического анализа

## 7.1 Общие сведения

Индикаторы технического анализа представляют собой отдельный класс скриптов, которые удовлетворяют определенным условиям и расположены в папке **LuaIndicators** в каталоге терминала. Если папка отсутствует в каталоге, необходимо создать ее вручную. Список скриптов недоступен из диалога **Сервисы / LUA скрипты....**

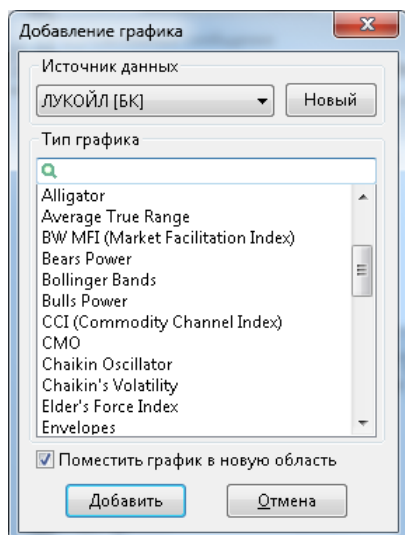
При добавлении нового индикатора на график плагин **qlua** сканирует папку **LuaIndicators**, проверяет файлы с расширением lua и luac (скомпилированные скрипты lua) на соответствие следующим требованиям:

- определена функция Init;
- определена функция OnCalculate;
- определена таблица Lua с именем Settings, в которой есть поле «Name».

Пример минимального корректного кода для индикатора:

```
Settings={}
Settings.Name = "minimal"
function Init()
    return 1
end
function OnCalculate(index)
    return 1
end
```

Список доступных индикаторов передается в модуль **qchart** и в дальнейшем доступен из стандартного диалога добавления индикатора на графике:



Список типов графиков отсортирован по алфавиту, за исключением типов графиков «Price» и «Volume», которые всегда располагаются в начале списка.

## 7.2 Функции и глобальные переменные скрипта индикатора

### 7.2.1 Init

Функция вызывается при добавлении индикатора на график (при нажатии кнопки «Добавить» в окне «Добавление графика»). Возвращает число, которое определяет количество линий в индикаторе.

**Функция вызывается также при перезагрузке Рабочего места QUIK и при загрузке wnd-файла либо tab-файла, в которых сохранен график с индикатором.**

Формат вызова:

NUMBER Init()

Например, для индикатора «Аллигатор» функция возвращает значение «3».

### 7.2.2 OnCalculate

Функция вызывается при поступлении новой или изменении существующей свечки в источнике данных для индикатора.

Формат вызова:

NUMBER v<sub>1</sub> [, NUMBER v<sub>n</sub>] OnCalculate(NUMBER index)

Параметры:

- **index** – индекс свечи в источнике данных. Начинается с «1».

Если значение  $v_i$  не определено, то функция возвращает nil в качестве значения линии на интервале index.

Пример:

```
function Init()
    myDEMA = cached_DTEMA()
    return 2
end
function OnCalculate(index)
    x, y = myDEMA(index, Settings.period, Settings.calc_mode) --exponential
    return x, y
end
```

### 7.2.3 OnDestroy

Функция вызывается при удалении индикатора с графика, либо при закрытии окна диаграммы и не является обязательной для индикатора.

Формат вызова:

OnDestroy ()

### 7.2.4 Функции для доступа к источнику данных

- Функции для доступа к источнику данных **O, H, L, C, V, T** принимают в качестве параметра индекс свечи и возвращают соответствующее значение в формате:

NUMBER <название функции>(NUMBER index)

- Функция **Size** возвращает текущее количество свечек в источнике данных. Формат функции:

NUMBER Size()

Описание значений функций **O, H, L, C, V, T, Size** совпадает со значениями, приведенными в п. [3.10.4](#).

Пример скрипта, реализующего индикатор «Moving Average»:

```
Settings={}
Settings.Name = "SimpleMA"
Settings.mode = "C"
Settings.period = 5
Settings.str_field = "STRING field"
```

```

function dValue(i,param)
    local v = param or "C"
    if v == "O" then
        return O(i)
    elseif v == "H" then
        return H(i)
    elseif v == "L" then
        return L(i)
    elseif v == "C" then
        return C(i)
    elseif v == "V" then
        return V(i)
    elseif v == "M" then
        return (H(i) + L(i))/2
    elseif v == "T" then
        return (H(i) + L(i)+C(i))/3
    elseif v == "W" then
        return (H(i) + L(i)+2*C(i))/4
    else
        return C(i)
    end
end

end

function Init()
    return 1
end

function OnCalculate(idx)
    local per = Settings.period
    local mode = Settings.mode
    local lValue = iValue
    if idx >= per then
        local ma_value=0
        for j = (idx-per)+1, idx do
            ma_value = ma_value+dValue(j, mode)
        end
        return ma_value/per
    else
        return nil
    end
end

end

```

### 7.2.5 getSourceInfo

Функция предназначена для получения информации об источнике данных для индикатора.

TABLE info getSourceInfo()

Функция возвращает таблицу Lua с параметрами:

**ВАЖНО!** Для корректной работы функции getSourceInfo, вызываемой из функции Init, необходимо перезапустить Рабочее место QUIK после добавления индикатора на график.

| Параметр   | Тип    | Описание  |
|------------|--------|---|
| interval   | NUMBER | Текущий интервал (тайм-фрейм) графика   |
| class_code | STRING | Код класса источника данных   |
| sec_code   | STRING | Код бумаги источника данных   |
| param      | STRING | Наименование параметра Таблицы текущих торгов, по которому строится график. Если поле пустое, то график строится на основании Таблицы обезличенных сделок |

Возможные значения поля interval:

| Возвращаемое значение | Интервал |
|-----------------------|----------|
| 0                     | Тиковый  |
| 1                     | 1 минута |
| 2                     | 2 минуты |
| 3                     | 3 минуты |
| 4                     | 4 минуты |
| 5                     | 5 минут  |
| 6                     | 6 минут  |
| 10                    | 10 минут |
| 15                    | 15 минут |

| Возвращаемое значение | Интервал |
|-----------------------|----------|
| 20                    | 20 минут |
| 30                    | 30 минут |
| 60                    | 1 час    |
| 120                   | 2 часа   |
| 240                   | 4 часа   |
| -1                    | 1 день   |
| -2                    | 1 неделя |
| -3                    | 1 месяц  |

### 7.2.6 SetValue

Функция предназначена для установки указанного значения на выбранной линии определенной свечи индикатора:

BOOLEAN SetValue(NUMBER index, NUMBER line\_number, NUMBER value)

Параметры:

- **index** – индекс свечи;
- **line\_number** – номер линии;
- **value** – устанавливаемое значение.

Функция возвращает «true» в случае успешного завершения, иначе – «false».

Пример использования функции приведен ниже.

### 7.2.7 GetValue

Функция предназначена для определения значения, установленного на выбранной линии указанной свечи индикатора:

NUMBER value GetValue(NUMBER index, NUMBER line\_number)

Параметры:

- **index** – индекс свечи;
- **line\_number** – номер линии.

Функция возвращает значение **value**, установленное для линии **line\_number** свечи **index**. В случае ошибки функция возвращает «nil».

Пример:

```
function OnCalculate(i)
    local ret_value = 0
    if i == 1 then
        ret_value = 1
    else
        ret_value = GetValue(i-1, 1)+2
    end
    if i%3 == 0 then
        ret_value = SetValue(i-1, 1, 2)
    end
    return ret_value
end
```

### 7.2.8 SetRangeValue

Функция предназначена для установки указанного значения на выбранной линии для определенного интервала индексов свечей индикатора:

```
BOOLEAN SetRangeValue(NUMBER line_number, NUMBER start_index, NUMBER  
end_index, NUMBER value)
```

Параметры:

- **line\_number** – номер линии;
- **start\_index** – индекс начальной свечи интервала;
- **end\_index** – индекс конечной свечи интервала;
- **value** – устанавливаемое значение.

Функция устанавливает значение **value** для линии **line\_number** от индекса **start\_index** до индекса **end\_index** включительно.

Функция возвращает «true» в случае успешного завершения, иначе – «false».

Пример:

```
function OnCalculate(index)  
    local range = Settings.range  
    if index >= range then  
        SetValue(index-range, 1, nil)  
        SetValue(index-range, 2, nil)  
  
        SetValue(index-range+1, 1, H(index-range+1))  
        SetValue(index-range+1, 2, L(index-range+1))  
        SetRangeValue(1, index-range+2, index-1, nil)  
        SetRangeValue(2, index-range+2, index-1, nil)  
  
        --[[  
        for i = index-range+2, index-1 do  
            SetValue(i, 1, nil)  
            SetValue(i, 2, nil)  
        end  
        --]]  
  
        return H(index), L(index)  
    else  
        return nil, nil  
    end  
end
```



## 7.2.9 Таблица «Settings»

Таблица **Settings** содержит настройки индикатора и в скрипте объявляется как глобальная.

Список предопределённых полей с примерами:

- **STRING Name** – строка с названием индикатора.

```
Settings.Name = "Two MA"
```

- **STRING line[n].Name** – строка с именем линии с номером N. Индексы линий начинаются с «1».

```
Settings.line[1].Name = "First MA"  
Settings.line[2].Name = "Second MA"
```

- **NUMBER line[n].Type** – тип отображения линии. Задаются с помощью предопределённых констант: TYPE\_LINE, TYPE\_DASH, TYPE\_DOT, TYPE\_HISTOGRAM, TYPE\_TRIANGLE\_UP, TYPE\_TRIANGLE\_DOWN.

```
Settings.line[1].Type = TYPE_LINE --линии  
Settings.line[2].Type = TYPE_DASH --тире  
Settings.line[3].Type = TYPE_DOT --точки
```

- **NUMBER line[n].Width** – толщина линии.

```
Settings.line[1].Width = 5
```

- **NUMBER line[n].Color** – цвет линии. Результат выполнения функции RGB.

```
Settings.line[1].Color = RGB(255, 0, 0)  
Settings.line[2].Color = RGB(0, 255, 0)
```

Поля в таблице **Settings** отображаются в диалоге настроек в разделе «Пользовательские настройки».

Типы пользовательских параметров: числа и строки.

Поля, значения которых не определены в скрипте, будут проинициализированы значениями по умолчанию.

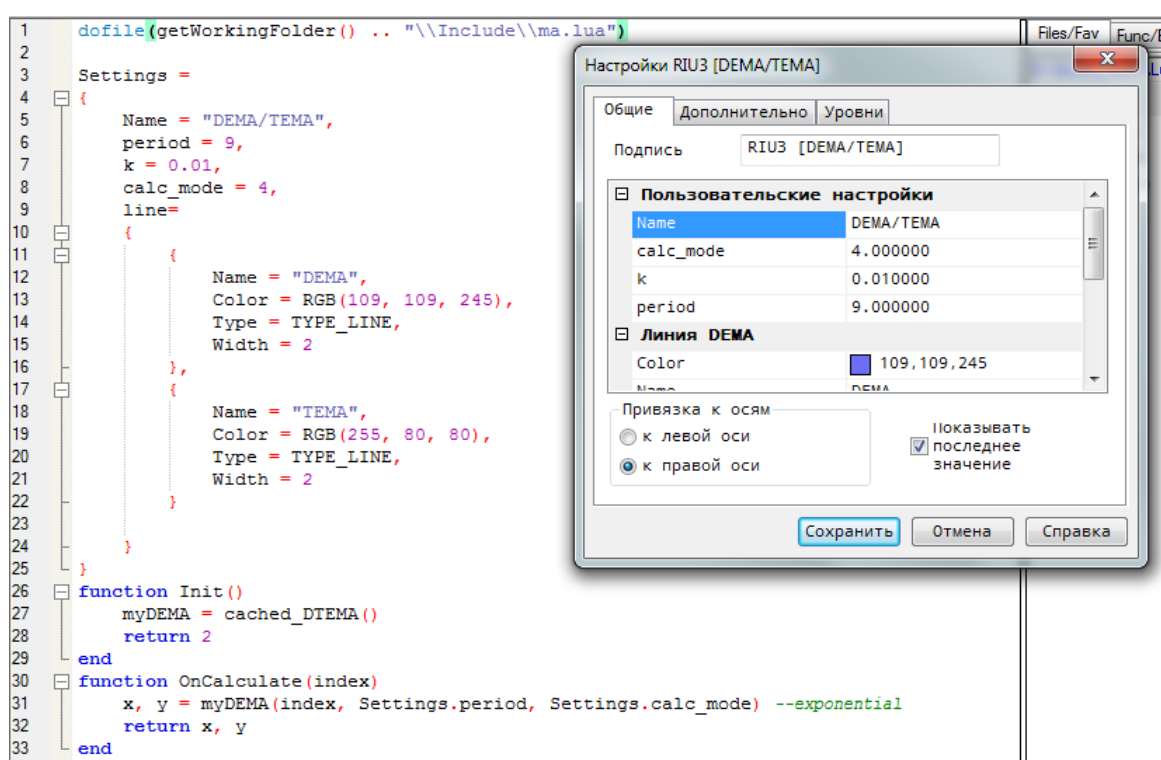
Для параметров `Settings.Name`, `Settings.line[n].Name` (и любых других пользовательских строковых параметров) не рекомендуется использовать многострочную конструкцию.

В случае использования многострочной конструкции, считается только первая строка, например, для параметра вида:

```
Settings.Name = [[Two  
MA]]
```

названием индикатора станет строка «Two».

### 7.2.10 Пример диалога настроек с привязкой таблицы «Settings»



Изменение настроек в диалоге приводит к изменению значений полей таблицы `Settings` на работающей Lua-машине без изменений исходного кода.

### 7.2.11 Список функций, доступных из скрипта индикатора

- **getWorkingFolder** – возвращает путь, по которому находится файл `info.exe`, исполняющий скрипт,
- **getScriptPath** – возвращает путь, по которому находится запускаемый скрипт,
- **getNumberOf** – возвращает количество записей в таблице «TableName»,
- **getItem** – возвращает таблицу Lua, содержащую информацию о данных из строки с номером «Index» из таблицы с именем «TableName»,

- **getParamEx** – получает значения всех параметров биржевой информации из Таблицы текущих торгов,
- **message** – отображает сообщения в терминале QUIK,
- **isConnected** – определяет состояние подключения клиентского места к серверу,
- **getTradeDate** – получает дату торговой сессии,
- **getInfoParam** – позволяет получить параметры для информационного окна (Связь/Информационное окно),
- **getClassSecurities** – получает список кодов бумаг для списка классов, заданного списком кодов,
- **getClassInfo** – получает информацию о классе,
- **getClassesList** – получает список кодов классов, полученных с сервера в ходе сеанса связи,
- **getSecurityInfo** – получает информацию по инструменту,
- **getQuoteLevel2** – получает стакан по указанному классу и бумаге,
- **getMoney** – получает информацию по денежным лимитам,
- **getDepo** – получает информацию по бумажным лимитам,
- **sendTransaction** – функция для работы с заявками,
- **SearchItems** – позволяет реализовать быструю выборку элементов из хранилища терминала и возвращает таблицу с индексами элементов, удовлетворяющих условию поиска,
- **getPortfolioInfo** – получает значения параметров таблицы «Клиентский портфель»,
- **getBuySellInfo** – получает параметры таблицы «Купить/Продать»,
- **getPortfolioInfoEx** – получает значения параметров таблицы «Клиентский портфель» с учетом вида лимита,
- **getBuySellInfoEx** – получает параметры таблицы «Купить/Продать» с учетом вида лимита,
- **getOrderByNumber** – возвращает таблицу Lua, содержащую описание параметров Таблицы заявок и индекс заявки в хранилище терминала,
- **RGB** – преобразовывает компоненты RGB (red, green, blue) в одно число.
- **AddLabel** – добавляет метку с заданными параметрами.
- **DelLabel** – удаляет метку с заданными параметрами.
- **DelAllLabels** – удаляет все метки на диаграмме с указанным графиком.
- **GetLabelParams** – получает параметры метки.
- **SetLabelParams** – задает параметры для метки с указанным идентификатором.
- **SetValue** – устанавливает указанное значение на выбранной линии определенной свечи индикатора.
- **GetValue** – получает значение, установленное на выбранной линии указанной свечи индикатора.
- **SetRangeValue** – устанавливает указанное значение на выбранной линии для определенного интервала индексов свечей индикатора.

### 7.2.12 Загрузка и сохранение настроек индикатора в файл

При выборе пункта меню **Система / Сохранить настройки в файл...**, сохраняются все значения из таблицы **Settings** в wnd-файл.

При загрузке настроек из файла, модуль **qchart** получает от модуля **qlua** список индикаторов и автоматически создает индикатор по его имени.

Если загружаемый индикатор отсутствует в списке (например, удален файл или изменялось значение в поле «Settings.Name»), индикатор не отображается. Для того чтобы индикаторы не пропадали при изменении настроек в коде Lua, на графике отображается его легенда и доступен диалог редактирования настроек.

## 8. Потокобезопасные функции для работы с таблицами Lua

Одновременная работа с таблицами из функций обратного вызова скрипта и функции `main()` может приводить к неопределенным ситуациям. Для решения этой проблемы `qlua.dll` предоставляет потокобезопасные аналоги стандартных функций Lua.

**Выполнение потокобезопасной функции блокирует выполнение кода в другом потоке до окончания работы функции.**

Формат вызова потокобезопасной функции совпадает с форматом вызова аналогичной стандартной функции Lua.

В таблице представлены стандартные функции Lua и соответствующие им потокобезопасные аналоги:

| Стандартная функция Lua | Потокобезопасная функция |
|-------------------------|--------------------------|
| concat                  | sconcat                  |
| remove                  | sremove                  |
| insert                  | sinsert                  |
| sort                    | ssort                    |

## 9. Приложения

### Приложение 1. Пример скрипта на языке Lua

В данном приложении приведен пример скрипта на языке Lua для создания таблицы в Рабочем месте QUIK.

#### Файл `table_object.lua`

```
dofile (getScriptPath() .. "\\quik_table_wrapper.lua")
dofile (getScriptPath() .. "\\ntime.lua")
stopped = false
function format1(data)
    return string.format("0x%08X", data)
end

function format2(data)
    return string.format("%06d", data)
end

function OnStop(s)
    stopped = true
end

function main()
    -- поворачивающиеся «палочки» в заголовке таблицы
    local palochki = {"-", "\\ ", "|", "/" }
    -- создать экземпляр QTable
    t = QTable.new()
    if not t then
        message("error!", 3)
        return
    else
        message("table with id = " .. t.t_id .. " created", 1)
    end

    -- добавить два столбца с функциями форматирования
    -- в первом столбце - hex-значения, во втором - целые числа
    t.AddColumn("test1", QTABLE_INT_TYPE, 10, format1)
    t.AddColumn("test2", QTABLE_INT_TYPE, 10, format2)
    -- добавить столбцы без форматирования
    t.AddColumn("test3", QTABLE_CACHED_STRING_TYPE, 50)
    t.AddColumn("test4", QTABLE_TIME_TYPE, 50)
    t.AddColumn("test5", QTABLE_CACHED_STRING_TYPE, 50)
```

```

t:SetCaption("Test")
t:Show()
i=1
-- исполнять цикл, пока пользователь не остановит скрипт из диалога управления
while not stopped do
    -- если таблица закрыта, то показать ее заново
    -- при этом все предыдущие данные очищаются
    if t:IsClosed() then
        t:Show()
    end
    -- на каждой итерации повернуть «палочку» на 45 градусов
    t:SetCaption("QLUA TABLE TEST " .. palochki[i%4 +1])
    -- метод добавит в таблицу новую строку и вернет ее номер
    local row = t:AddLine()
    t:SetValue(row, "test1", row, i)
    t:SetValue(row, "test2", row, i)

    -- заполнить ячейку текущим заголовком таблицы
    -- тип столбца - строковый, поэтому последний параметр пропускается
    SetCell(t.t_id, row, 3, GetWindowCaption(t.t_id))

    _date = os.date("*t")
    -- 4-й столбец заполнить данными типа время (число в формате <ЧЧММСС>)
    -- Функция для строкового представления времени определена в файле
ntime.lua
    -- Функция NiceTime возвращает строку
    SetCell(t.t_id, row, 4,
        NiceTime(_date) .. string.format(" (%02d:%02d:%02d)", _date.hour,
_date.min, _date.sec),
        _date.hour*10000+_date.min*100 + _date.sec)
    -- пятый столбец имеет строковый тип и заполняется результатом
выполнения функции NiceTime
    -- исходный код функции взят из виджета Conky Lua для Ubuntu
    SetCell(t.t_id, row, 5, NiceTime(_date))
    sleep(1000)
    i=i+1
end
message("finished")
end

```

### Файл quik\_table\_wrapper.lua

```

-- Перегрузка функции message с необязательным вторым параметром
old_message = message
function message(v, i)
    old_message(tostring(v), i or 1)
end

```

```

end

QTable = {}
QTable.__index = QTable

-- Создать и инициализировать экземпляр таблицы QTable
function QTable.new()
    local t_id = AllocTable()
    if t_id ~= nil then
        q_table = {}
        setmetatable(q_table, QTable)
        q_table.t_id=t_id
        q_table.caption = ""
        q_table.created = false
        q_table.curr_col=0
        -- таблица с описанием параметров столбцов
        q_table.columns={}
        return q_table
    else
        return nil
    end
end

function QTable:Show()
    -- отобразить в терминале окно с созданной таблицей
    CreateWindow(self.t_id)
    if self.caption ~= "" then
        -- задать заголовок для окна
        SetWindowCaption(self.t_id, self.caption)
    end
    self.created = true
end

function QTable:IsClosed()
    -- если окно с таблицей закрыто, возвращает «true»
    return IsWindowClosed(self.t_id)
end

function QTable:delete()
    -- удалить таблицу
    DestroyTable(self.t_id)
end

function QTable:GetCaption()
    if IsWindowClosed(self.t_id) then
        return self.caption
    end
end

```

```

        else
            -- возвращает строку, содержащую заголовок таблицы
            return GetWindowCaption(self.t_id)
        end
    end
end

-- Задать заголовок таблицы
function QTable:SetCaption(s)
    self.caption = s
    if not IsWindowClosed(self.t_id) then
        res = SetWindowCaption(self.t_id, tostring(s))
    end
end

-- Добавить описание столбца <name> типа <c_type> в таблицу
-- <ff> - функция форматирования данных для отображения
function QTable:AddColumn(name, c_type, width, ff )
    local col_desc={}
    self.curr_col=self.curr_col+1
    col_desc.c_type = c_type
    col_desc.format_function = ff
    col_desc.id = self.curr_col
    self.columns[name] = col_desc
    -- <name> используется в качестве заголовка таблицы
    AddColumn(self.t_id, self.curr_col, name, true, c_type, width)
end

function QTable:Clear()
    -- очистить таблицу
    Clear(self.t_id)
end

-- Установить значение в ячейке
function QTable:SetValue(row, col_name, data)
    local col_ind = self.columns[col_name].id or nil
    if col_ind == nil then
        return false
    end
    -- если для столбца задана функция форматирования, то она используется
    local ff = self.columns[col_name].format_function

    if type(ff) == "function" then
        -- в качестве строкового представления используется
        -- результат выполнения функции форматирования
        SetCell(self.t_id, row, col_ind, ff(data), data)
    end
    return true
end

```



```

        else
            SetCell(self.t_id, row, col_ind, tostring(data), data)
        end
    end
end

function QTable:AddLine()
    -- добавляет в конец таблицы пустую строку и возвращает ее номер
    return InsertRow(self.t_id, -1)
end

function QTable:GetSize()
    -- возвращает размер таблицы
    return GetTableSize(self.t_id)
end

-- Получить данные из ячейки по номеру строки и имени столбца
function QTable:GetValue(row, name)
    local t={}
    local col_ind = self.columns[name].id
    if col_ind == nil then
        return nil
    end
    t = GetCell(self.t_id, row, col_ind)
    return t
end

-- Задать координаты окна
function QTable:SetPosition(x, y, dx, dy)
    return SetWindowPos(self.t_id, x, y, dx, dy)
end

-- Функция возвращает координаты окна
function QTable:GetPosition()
    top, left, bottom, right = GetWindowRect(self.t_id)
    return top, left, right-left, bottom-top
end

```

## Файл ntime.lua

```

words = {"one ", "two ", "three ", "four ", "five ", "six ", "seven ", "eight ",
"nine "}
levels = {"thousand ", "million ", "billion ", "trillion ", "quadrillion ",
"quintillion ", "sextillion ", "septillion ", "octillion ", [0] = ""}
iwords = {"ten ", "twenty ", "thirty ", "forty ", "fifty ", "sixty ", "seventy ",
"eighty ", "ninety "}

```

```

twords = {"eleven ", "twelve ", "thirteen ", "fourteen ", "fifteen ", "sixteen ",
"seventeen ", "eighteen ", "nineteen "}

function digits(n)
    local i, ret = -1
    return function()
        i, ret = i + 1, n % 10
        if n > 0 then
            n = math.floor(n / 10)
            return i, ret
        end
    end
end

level = false
function getname(pos, dig)
    level = level or pos % 3 == 0
    if(dig == 0) then return "" end
    local name = (pos % 3 == 1 and iwords[dig] or words[dig]) .. (pos % 3 == 2 and
"hundred " or "")
    if(level) then name, level = name .. levels[math.floor(pos / 3)], false end
    return name
end

function numberToWord(number)
    if(number == 0) then return "zero" end
    vword = ""
    for i, v in digits(number) do
        vword = getname(i, v) .. vword
    end

    for i, v in ipairs(words) do
        vword = vword:gsub("ty " .. v, "ty-" .. v)
        vword = vword:gsub("ten " .. v, twords[i])
    end
    return vword
end

function _Time(t)
    hour = t.hour
    minute = t.min
    hour = hour % 12
    if(hour == 0) then
        hour, nextHourWord = 12, "one "
    else
        nextHourWord = numberToWord(hour+1)
    end
end

```

```

end
hourWord = numberToWord(hour)
if(minute == 0 ) then
return hourWord .. "o'clock"
elseif(minute == 30) then
return "half past " .. hourWord
elseif(minute == 15) then
return "a quarter past " .. hourWord
elseif(minute == 45) then
return "a quarter to " .. nextHourWord
else
if(minute < 30) then
return numberToWord(minute) .. "past " .. hourWord
else
return numberToWord(60-minute) .. "to " .. nextHourWord
end
end
end

function _Seconds(s)
return numberToWord(s)
end

function NiceTime(t)
return _Time(t) .. "and " .. _Seconds(t.sec) .. "second"
end

```

В результате выполнения скрипта в Рабочем месте QUIK создается таблица вида:

|    | test1      | test2  | test3             | test4   | test5                                |
|----|------------|--------|-------------------|---|--------------------------------------|
| 1  | 0x00000001 | 000001 | QLUA TABLE TEST \ | twelve to five and eleven second (16:48:11)     | twelve to five and eleven second     |
| 2  | 0x00000002 | 000002 | QLUA TABLE TEST   | twelve to five and twelve second (16:48:12)     | twelve to five and twelve second     |
| 3  | 0x00000003 | 000003 | QLUA TABLE TEST / | twelve to five and thirteen second (16:48:13)   | twelve to five and thirteen second   |
| 4  | 0x00000004 | 000004 | QLUA TABLE TEST - | twelve to five and fourteen second (16:48:14)   | twelve to five and fourteen second   |
| 5  | 0x00000005 | 000005 | QLUA TABLE TEST \ | twelve to five and fifteen second (16:48:15)    | twelve to five and fifteen second    |
| 6  | 0x00000006 | 000006 | QLUA TABLE TEST   | twelve to five and sixteen second (16:48:16)    | twelve to five and sixteen second    |
| 7  | 0x00000007 | 000007 | QLUA TABLE TEST / | twelve to five and seventeen second (16:48:17)  | twelve to five and seventeen second  |
| 8  | 0x00000008 | 000008 | QLUA TABLE TEST - | twelve to five and eighteen second (16:48:18)   | twelve to five and eighteen second   |
| 9  | 0x00000009 | 000009 | QLUA TABLE TEST \ | twelve to five and nineteen second (16:48:19)   | twelve to five and nineteen second   |
| 10 | 0x0000000A | 000010 | QLUA TABLE TEST   | twelve to five and twenty second (16:48:20)     | twelve to five and twenty second     |
| 11 | 0x0000000B | 000011 | QLUA TABLE TEST / | twelve to five and twenty-one second (16:48:21) | twelve to five and twenty-one second |
| 12 | 0x0000000C | 000012 | QLUA TABLE TEST - | twelve to five and twenty-two second (16:48:22) | twelve to five and twenty-two second |

## Приложение 2. Примеры сортировки в таблицах

Примеры сортировки в столбце таблицы, содержащем данные числового (столбец «test4») и строкового (столбец «test5») типа:

|    | test1      | test2  | test3             | test4   | test5                                |
|----|------------|--------|-------------------|---|--------------------------------------|
| 1  | 0x00000001 | 000001 | QLUA TABLE TEST \ | twelve to five and eleven second (16:48:11)     | twelve to five and eleven second     |
| 2  | 0x00000002 | 000002 | QLUA TABLE TEST   | twelve to five and twelve second (16:48:12)     | twelve to five and twelve second     |
| 3  | 0x00000003 | 000003 | QLUA TABLE TEST / | twelve to five and thirteen second (16:48:13)   | twelve to five and thirteen second   |
| 4  | 0x00000004 | 000004 | QLUA TABLE TEST - | twelve to five and fourteen second (16:48:14)   | twelve to five and fourteen second   |
| 5  | 0x00000005 | 000005 | QLUA TABLE TEST \ | twelve to five and fifteen second (16:48:15)    | twelve to five and fifteen second    |
| 6  | 0x00000006 | 000006 | QLUA TABLE TEST   | twelve to five and sixteen second (16:48:16)    | twelve to five and sixteen second    |
| 7  | 0x00000007 | 000007 | QLUA TABLE TEST / | twelve to five and seventeen second (16:48:17)  | twelve to five and seventeen second  |
| 8  | 0x00000008 | 000008 | QLUA TABLE TEST - | twelve to five and eighteen second (16:48:18)   | twelve to five and eighteen second   |
| 9  | 0x00000009 | 000009 | QLUA TABLE TEST \ | twelve to five and nineteen second (16:48:19)   | twelve to five and nineteen second   |
| 10 | 0x0000000A | 000010 | QLUA TABLE TEST   | twelve to five and twenty second (16:48:20)     | twelve to five and twenty second     |
| 11 | 0x0000000B | 000011 | QLUA TABLE TEST / | twelve to five and twenty-one second (16:48:21) | twelve to five and twenty-one second |
| 12 | 0x0000000C | 000012 | QLUA TABLE TEST - | twelve to five and twenty-two second (16:48:22) | twelve to five and twenty-two second |

|    | test1      | test2  | test3             | test4   | test5                                |
|----|------------|--------|-------------------|---|--------------------------------------|
| 1  | 0x00000008 | 000008 | QLUA TABLE TEST - | twelve to five and eighteen second (16:48:18)   | twelve to five and eighteen second   |
| 2  | 0x00000001 | 000001 | QLUA TABLE TEST \ | twelve to five and eleven second (16:48:11)     | twelve to five and eleven second     |
| 3  | 0x00000005 | 000005 | QLUA TABLE TEST \ | twelve to five and fifteen second (16:48:15)    | twelve to five and fifteen second    |
| 4  | 0x00000004 | 000004 | QLUA TABLE TEST - | twelve to five and fourteen second (16:48:14)   | twelve to five and fourteen second   |
| 5  | 0x00000009 | 000009 | QLUA TABLE TEST \ | twelve to five and nineteen second (16:48:19)   | twelve to five and nineteen second   |
| 6  | 0x00000007 | 000007 | QLUA TABLE TEST / | twelve to five and seventeen second (16:48:17)  | twelve to five and seventeen second  |
| 7  | 0x00000006 | 000006 | QLUA TABLE TEST   | twelve to five and sixteen second (16:48:16)    | twelve to five and sixteen second    |
| 8  | 0x00000003 | 000003 | QLUA TABLE TEST / | twelve to five and thirteen second (16:48:13)   | twelve to five and thirteen second   |
| 9  | 0x00000002 | 000002 | QLUA TABLE TEST   | twelve to five and twelve second (16:48:12)     | twelve to five and twelve second     |
| 10 | 0x0000000A | 000010 | QLUA TABLE TEST   | twelve to five and twenty second (16:48:20)     | twelve to five and twenty second     |
| 11 | 0x0000000B | 000011 | QLUA TABLE TEST / | twelve to five and twenty-one second (16:48:21) | twelve to five and twenty-one second |
| 12 | 0x0000000C | 000012 | QLUA TABLE TEST - | twelve to five and twenty-two second (16:48:22) | twelve to five and twenty-two second |

## Приложение 3. Примеры обработки событий для таблиц

### Пример обработки событий мыши и клавиатуры

```
stopped = false
t_id = nil

old_message = message
local fmt = string.format
function message(v, t)
    t= t or 1
    old_message(tostring(v), t)
end

function OnStop(s)
    stopped = true
    if t_id~= nil then
        DestroyTable(t_id)
    end
end

event_table = {
    [QTABLE_LBUTTONDOWN] = "Нажали левую кнопку мыши",
    [QTABLE_RBUTTONDOWN] = "Нажали правую кнопку мыши",
    [QTABLE_LBUTTONDBLCLK] = "Левый даблклик",
    [QTABLE_RBUTTONDBLCLK] = "Правый даблклик",
    [QTABLE_SELCHANGED] = "Изменилась строка",
    [QTABLE_CHAR] = "Символьная клавиша",
    [QTABLE_VKEY] = "Еще какая-то клавиша",
    [QTABLE_CONTEXTMENU] = "Контекстное меню",
    [QTABLE_MBUTTONDOWN] = "Нажали на колесико мыши",
    [QTABLE_MBUTTONDBLCLK] = "Даблклик колесом",
    [QTABLE_LBUTTONUP] = "Отпустили левую кнопку мыши",
    [QTABLE_RBUTTONUP] = "Отпустили правую кнопку мыши",
    [QTABLE_CLOSE] = "Закрыли таблицу"
}

function event_callback_str(t_id, msg, par1, par2)
    local str = fmt("%s, par1 = %d, par2 = %d", event_table[msg], par1, par2)
    SetWindowCaption(t_id, str)
    message(str)
end

local p_row = -1
local p_col = -1
function event_callback_color(t_id, msg, par1, par2)
    if par1==3 and par2 == 1 then
```

```

        os.exit()
    end
    if msg == QTABLE_LBUTTONDOWN then
        if p_col ~= -1 and p_col ~= -1 then
            SetColor(t_id, p_row, p_col, QTABLE_DEFAULT_COLOR,
QTABLE_DEFAULT_COLOR, QTABLE_DEFAULT_COLOR)
            end
            SetColor(t_id, par1, par2, RGB(240, 128, 128), QTABLE_DEFAULT_COLOR,
QTABLE_DEFAULT_COLOR, QTABLE_DEFAULT_COLOR)
            p_row = par1
            p_col = par2
        end
    end
end

function main()

    data = {
        {"1", 2, 20130530},
        {"4", 5, 20130529},
        {"7", 8, 20130528}
    }

    t_id = AllocTable()
    message (t_id)
    AddColumn(t_id, 1, "строка", true, QTABLE_CACHED_STRING_TYPE, 10)
    AddColumn(t_id, 2, "число", true, QTABLE_INT_TYPE, 10)
    AddColumn(t_id, 3, "дата", true, QTABLE_DATE_TYPE, 10)
    CreateWindow(t_id)

    for _, v in pairs(data) do
        row = InsertRow(t_id, -1)
        SetCell(t_id, row, 1, v[1])
        SetCell(t_id, row, 2, string.format("value = %d",v[2]), v[2])
        SetCell(t_id, row, 3, string.format("%04d - %02d - %02d",v[3]/10000,
(v[3]%10000)/100, v[3]%100), v[3])
    end
    SetWindowCaption(t_id, "EXAMPLE")
    SetTableNotificationCallback(t_id, event_callback_str)
    sleep(5000)
    SetTableNotificationCallback(t_id, event_callback_color)
    SetCell(t_id, 3, 1, "DO NOT CLICK ME")
    SetTableNotificationCallback(t_id, dummy)

    while not stopped do
        sleep(100)
    end
end

```

```
end
end
```

## Пример реализации игры «Крестики-нолики»

```
--[[ TIC-TAC-TOE
by Evan Hahn (http://evanhahn.com/how-to-code-tic-tac-toe-and-a-lua-implementation/)
--]]

-----
-- Configuration (change this if you wish!) --
-----

t_id=nil --grid
-- Are they playable by human or computer-controlled?
PLAYER_1_HUMAN = true
PLAYER_2_HUMAN = false

-- Board size
BOARD_RANK = 3      -- The board will be this in both dimensions.

-- Display stuff
PLAYER_1 = "[x]"     -- Player 1 is represented by this. Player 1 goes first.
PLAYER_2 = "[o]"     -- Player 2 is represented by this.
EMPTY_SPACE = "[ ]" -- An empty space is displayed like this.
DISPLAY_HORIZONTAL_SEPARATOR = "-"      -- Horizontal lines look like this.
DISPLAY_VERTICAL_SEPARATOR = " | "     -- Vertical lines look like this

--[[ #####
    Don't mess with things below here unless you are brave  #####
##### --]]

-----
-- More configuration --
-----

MAX_BOARD_RANK = 100-- Won't run above this number. Prevents crashes.

-----
-- Don't run if the board is larger than the maximum --
-----

if BOARD_RANK > MAX_BOARD_RANK then os.exit(0) end

-----
-- Create board (2D table) --
```

```

-----

space = {}
for i = 0, (BOARD_RANK - 1) do
    space[i] = {}
    for j = 0, (BOARD_RANK - 1) do
        space[i][j] = nil    -- start each space with nil
    end
end
end

-----

-- Board functions --
-----

-- get the piece at a given spot
function getPiece(x, y)
    return space[x][y]
end

-- get the piece at a given spot; if nil, return ""
-- this is useful for output.
function getPieceNoNil(x, y)
    if getPiece(x, y) ~= nil then
        return getPiece(x, y)
    else
        return EMPTY_SPACE
    end
end

-- is that space empty?
function isEmpty(x, y)
    if getPiece(x, y) == nil then
        return true
    else
        return false
    end
end

-- place a piece there, but make sure nothing is there already.
-- if you can't play there, return false.
function placePiece(x, y, piece)
    if isEmpty(x, y) == true then
        space[x][y] = piece
        return true
    else
        return false
    end
end

```



```

        end
    end

    -- is the game over?
    function isGameOver()
        if checkWin() == false then      -- if there is no win...
            for i = 0, (BOARD_RANK - 1) do    -- is the board empty?
                for j = 0, (BOARD_RANK - 1) do
                    if isEmpty(i, j) == true then return false end
                end
            end
            return true
        else    -- there is a win; the game is over
            return true
        end
    end
end

-- create a string made up of a certain number of smaller strings
-- this is useful for the display.
function repeatString(to_repeat, amount)
    if amount <= 0 then return "" end
    local to_return = ""
    for i = 1, amount do
        to_return = to_return .. to_repeat
    end
    return to_return
end

-- display the board.
-- this uses the configuration file pretty much entirely.
function displayBoard()
    for i = (BOARD_RANK - 1), 0, -1 do
        for j = 0, (BOARD_RANK - 1) do    -- generate that row
            local piece = getPieceNotNil(j, i)
            SetCell(t_id, i+1, j+1, piece)
        end
    end
end

-----
-- Create regions (I admit this is a bit ugly) --
-----

-- declare region and a number to increment
region = {}
region_number = 0

```

```

-- vertical
for i = 0, (BOARD_RANK - 1) do
    region[region_number] = {}
    for j = 0, (BOARD_RANK - 1) do
        region[region_number][j] = {}
        region[region_number][j]["x"] = i
        region[region_number][j]["y"] = j
    end
    region_number = region_number + 1
end

-- horizontal
for i = 0, (BOARD_RANK - 1) do
    region[region_number] = {}
    for j = 0, (BOARD_RANK - 1) do
        region[region_number][j] = {}
        region[region_number][j]["x"] = j
        region[region_number][j]["y"] = i
    end
    region_number = region_number + 1
end

-- diagonal, bottom-left to top-right
region[region_number] = {}
for i = 0, (BOARD_RANK - 1) do
    region[region_number][i] = {}
    region[region_number][i]["x"] = i
    region[region_number][i]["y"] = i
end
region_number = region_number + 1

-- diagonal, top-left to bottom-right
region[region_number] = {}
for i = (BOARD_RANK - 1), 0, -1 do
    region[region_number][i] = {}
    region[region_number][i]["x"] = BOARD_RANK - i - 1
    region[region_number][i]["y"] = i
end
region_number = region_number + 1

-----
-- Region functions --
-----

-- get a region

```

```

function getRegion(number)
    return region[number]
end

-- check for a win in a particular region.
-- returns a number representation of the region. occurrences of player 1
-- add 1, occurrences of player 2 subtract 1. so if there are two X pieces,
-- it will return 2. one O will return -1.
function checkWinInRegion(number)
    local to_return = 0
    for i, v in pairs(getRegion(number)) do
        local piece = getPiece(v["x"], v["y"])
        if piece == PLAYER_1 then to_return = to_return + 1 end
        if piece == PLAYER_2 then to_return = to_return - 1 end
    end
    return to_return
end

-- check for a win in every region.
-- returns false if no winner.
-- returns the winner if there is one.
function checkWin()
    for i in pairs(region) do
        local win = checkWinInRegion(i)
        if math.abs(win) == BOARD_RANK then
            if win == math.abs(win) then
                return PLAYER_1
            else
                return PLAYER_2
            end
        end
    end
    return false
end

-----
-- UI Functions --
-----

-- human play
function humanPlay(piece)
    message("Human turn")
    displayBoard()
    local placed = false
    while placed == false do    -- loop until they play correctly
        sleep(100)
    end
end

```

```

        if g_X ~= -1 and g_Y ~= -1 then
            local x = tonumber(g_Y)-1
            local y = tonumber(g_X)-1
            g_X = -1
            g_Y = -1
            message("clicked in " .. x .. " and " .. y)
            placed = placePiece(x, y, piece)
            if placed == false then
                message("I'm afraid you can't play there!")
            end
        end
    end
end
displayBoard()

end

-- AI play
function AIPlay(piece)

    -- am I negative or positive?
    local me = 0
    if piece == PLAYER_1 then me = 1 end
    if piece == PLAYER_2 then me = -1 end

    -- look for a region in which I can win
    for i in pairs(region) do
        local win = checkWinInRegion(i)
        if win == ((BOARD_RANK - 1) * me) then
            for j, v in pairs(getRegion(i)) do
                if isEmpty(v["x"], v["y"]) == true then
                    placePiece(v["x"], v["y"], piece)
                    return
                end
            end
        end
    end

    -- look for a region in which I can block
    for i in pairs(region) do
        local win = checkWinInRegion(i)
        if win == ((BOARD_RANK - 1) * (me * -1)) then
            for j, v in pairs(getRegion(i)) do
                if isEmpty(v["x"], v["y"]) == true then
                    placePiece(v["x"], v["y"], piece)
                    return
                end
            end
        end
    end
end

```

```

        end
    end
end

-- play first empty space, if no better option
for i = 0, (BOARD_RANK - 1) do
    for j = 0, (BOARD_RANK - 1) do
        if placePiece(i, j, piece) ~= false then return end
    end
end

end

end
g_X=-1
g_Y=-1
function event_callback(t_id, msg, par1, par2)
    if msg == QTABLE_LBUTTONDOWN then
        g_X = par1
        g_Y = par2
    end
end

old_message = message
local fmt = string.format
function message(v, t)
    t= t or 1
    old_message(tostring(v), t)
end

function main()
    t_id = AllocTable()
    AddColumn(t_id, 1, "", true, QTABLE_CACHED_STRING_TYPE, 5)
    AddColumn(t_id, 2, "", true, QTABLE_CACHED_STRING_TYPE, 5)
    AddColumn(t_id, 3, "", true, QTABLE_CACHED_STRING_TYPE, 5)
    CreateWindow(t_id)
    for i=1, 3 do
        row = InsertRow(t_id, -1)
        SetCell(t_id, row, 1, "[ ]")
        SetCell(t_id, row, 2, "[ ]")
        SetCell(t_id, row, 3, "[ ]")
    end
    SetTableNotificationCallback(t_id, event_callback)
    message("Welcome to Tic-Tac-Toe!")

    -- play the game until someone wins
    while true do
        sleep(100)
    end
end

```

```

-- break if the game is won
    if isGameOver() == true then
        break
    end
-- player 1
    if PLAYER_1_HUMAN == true then
        humanPlay(PLAYER_1)
    else
        AIPlay(PLAYER_1)
    end

    if isGameOver() == true then
        break
    end

    if PLAYER_2_HUMAN == true then
        humanPlay(PLAYER_2)
    else
        AIPlay(PLAYER_2)
    end
end

-- show the final board
displayBoard()

-- write who won, or if there is a tie
win = checkWin()
if win == false then
    message("Tie game!\n")
else
    message(win)
    message(" wins!\n")
end
end
end

```

## Приложение 4. Примеры использования параметра «params» в функции «SearchItems»

### Пример 1

Если не задан последний параметр в функции **SearchItems**, то в функцию обратного вызова **fn** передается обезличенная сделка в виде таблицы Lua:

```
function fn(t)
    if t.qty == 103 then
        return true
    else
        return false
    end
end
t1 = SearchItems("all_trades", 0, getNumberOf("all_trades")-1, fn)
```

### Пример 2

Если список полей задан, в функцию **fn** передаются параметры в том порядке, в котором они перечислены в списке параметров. В примере **par1** содержит поле **qty**, **par2** – **class\_code**, **par3** – **sec\_code**

Если перечисленные параметры отсутствуют в списке полей элемента, то в качестве параметра передается **nil**.

```
function fn(par1, par2, par3)
    if par1 == 103 and par2 == "SPBFUT" and par3 == "RIM3" then
        return true
    else
        return false
    end
end
t1 = SearchItems("all_trades", 0, getNumberOf("all_trades")-1, fn, "qty,class_code,sec_code")
```

### Пример 3

В примере **par1** имеет значение **nil**, **par2** – **class\_code**, **par3** – **sec\_code**:

```
function fn(par1, par2, par3)
    if par1 == 103 and par2 == "SPBFUT" and par3 == "RIM3" then
        return true
    else
        return false
    end
end
```

```
        end
    end
    t1 = SearchItems("all_trades", 0, getNumberOf("all_trades")-1, fn, "test,class_code,
    sec_code")
```

#### Пример 4

Элементы вложенных таблиц передаются через точку, например:

```
function fn(par1, par2)
    if par1 == 17 and par2 == 5 then
        return true
    else
        return false
    end
end
t1 = SearchItems("all_trades", 0, getNumberOf("all_trades")-1, fn, "datetime.hour,
datetime.min")
```