quik

Интерпретатор языка Lua

Руководство пользователя



Версия 13.2

2026

ARQA TECHNOLOGIES

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 22
   3.1 Функции для обращения к строкам произвольных таблиц QUIK 22
   3.2 Функции для обращения к спискам доступных параметров 25
   3.3 Функция для получения информации по денежным средствам 26
   3.4 Функция для получения позиций по инструментам 27
   3.5 Функция для получения информации по фьючерсным лимитам 28
   3.6 Функция для получения информации по фьючерсным позициям 28
   3.7 Функция для получения информации по инструменту 28
   3.8 Функция для получения даты торговой сессии 29
   3.9 Функция для получения стакана по указанному классу и инструменту 29
   3.10 Функции для работы с графиками 31
   3.11 Функции для работы с заявками 36
   3.12 Функции для получения значений таблицы «Текущие торги» 37
   3.13 Функции для получения параметров таблицы «Клиентский портфель» 40
   3.14 Функции для получения параметров таблицы «Купить/Продать» 47
   3.15 Функции для работы с таблицами Рабочего места QUIK 50
   3.16 Функции для работы с метками 58
   3.17 Функции для заказа стакана котировок 61
   3.18 Функции для заказа параметров Таблицы текущих торгов 62
   3.19 Функции для получения информации по единой денежной позиции 63
4. Структуры данных 64
   4.1 Классы 64

4.2 Фирмы 65
4.3 Обезличенные сделки 65
4.4 Сделки 66
4.5 Заявки 75
4.6 Позиции участника по торговым счетам 80
4.7 Позиции участника по инструментам 81
4.8 Стоп-заявки 81
4.9 Ограничения по клиентским счетам 83
4.10 Позиции по клиентским счетам (фьючерсы) 85
4.11 Позиции по деньгам 86
4.12 Удаление позиции по денежным средствам 87
4.13 Удаление позиции по инструментам 87
4.14 Удаление фьючерсного лимита 88
4.15 Позиции по инструментам 88
4.16 Позиции участника по деньгам 89
4.17 Заявки на внебиржевые сделки 90
4.18 Сделки для исполнения 94
4.19 Торговые счета 97
4.20 Отчеты по сделкам для исполнения 98
4.21 Инструменты 99
4.22 Свечки графика 101
4.23 Формат даты и времени, используемый в таблицах 102
4.24 Транзакции 102
4.25 Обязательства и требования по активам 106
4.26 Валюта: обязательства и требования по активам 106

5. Описание битовых флагов 107
5.1 Флаги для таблиц «Заявки», «Заявки на внебиржевые сделки» 107
5.2 Флаги для таблиц «Сделки», «Сделки для исполнения» 108
5.3 Флаги для таблицы «Отчеты по сделкам для исполнения» 108
5.4 Флаги для таблицы «Обезличенные сделки» 108
5.5 Флаги для таблицы «Стоп-заявки» 109

5.6 Дополнительные флаги для таблицы «Стоп-заявки» 109
5.7 Флаги для таблицы «Транзакции» 110
6. Функции для работы с битовыми масками в структурах данных 110
6.1 bit.tohex 110
6.2 bit.bnot 110
6.3 bit.band 111
6.4 bit.bor 111
6.5 bit.bxor 111
6.6 bit.test 111
7. Индикаторы технического анализа 112
7.1 Общие сведения 112
7.2 Функции и глобальные переменные скрипта индикатора 113
8. Потокобезопасные функции для работы с таблицами Lua 125
9. Приложения 126
Приложение 1. Пример скрипта на языке Lua 126
Приложение 2. Примеры сортировки в таблицах 133
Приложение 3. Примеры обработки событий для таблиц 134
Приложение 4. Примеры использования параметра «params» в функции «SearchItems» 144

Ваши пожелания и комментарии к данной Инструкции направляйте по электронной почте на адрес: 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.

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.

> 1. **Необходимо использовать 64-разрядные версии сторонних библиотек.**
> 2. **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

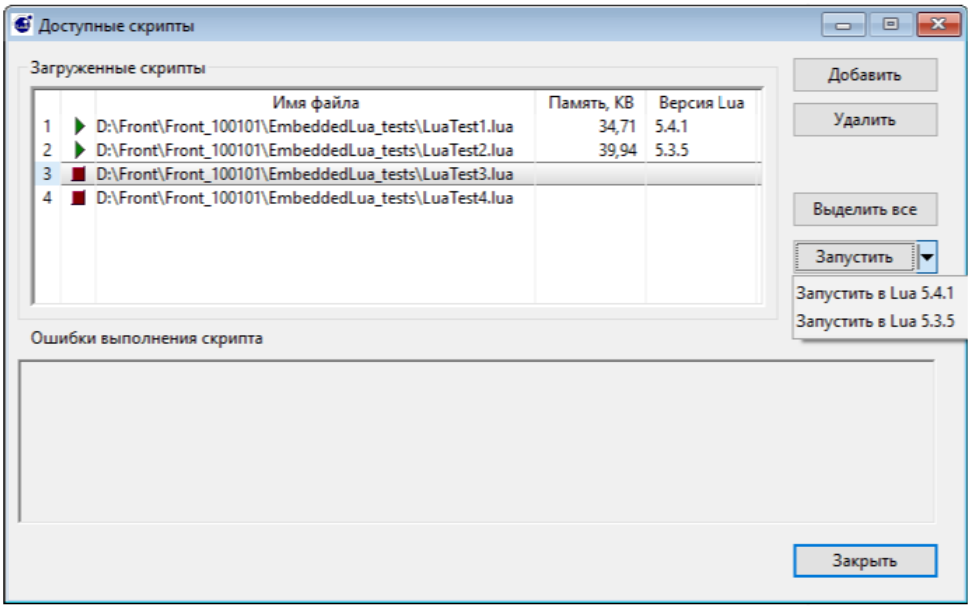
Доступ к функциям **QLua** осуществляется из главного меню программы. После установки компонента, в меню программы появляется дополнительный пункт **Сервисы / Lua скрипты**…

1.3.1 Назначение

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

1.3.2 Настройки окна

Настройки в этом окне:



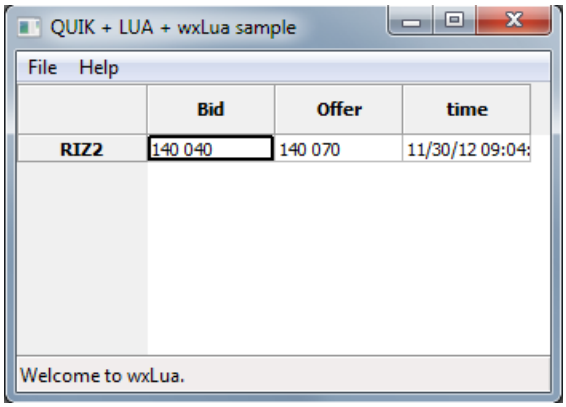
- «Добавить» – загрузить новый скрипт из файла;
- «Удалить» – удалить загруженный скрипт;
- «Выделить все» – выделить все доступные скрипты;
- «Запустить» – запустить выполнение загруженных скриптов, с возможностью выбрать версию Lua-машины**.** Возможные значения:
  - «Запустить в Lua 5.4.1»;
  - «Запустить в Lua 5.3.5».
- Значение по умолчанию при нажатии кнопки «Запустить» – «Запустить в Lua 5.4.1». Значение по умолчанию можно изменить в настройках Рабочего места QUIK в разделе «Lua скрипты» - «Версия Lua для скриптов пользователей», описание приведено в п. 2.10.7 Раздела 2 «Основные принципы работы».
- «Остановить» – остановить выполнение выбранного скрипта.

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

> **Скрипты запускаются автоматически после перезапуска терминала, если на момент закрытия терминала они были запущены.**

> **Значение в столбце «Память, KB» рассчитывается как сумма значений, возвращаемых функциями lua_gc(LUA_GCCOUNT) и lua_gc(LUA_GCCOUNTB). В скрипте Lua величину используемой памяти можно получить вызовом стандартной Lua-функции collectgarbage("count").**

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 слеша («\»). 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
LASTRECORDTIME	Время последней записи
NUMRECORDS	Число записей
LASTRECORD	Последняя запись
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

Значение параметра	Описание
MESSAGESSENT	Передано сообщений
ALLSENT	Передано всего байт
BYTESSENT	Передано полезных байт
BYTESPERSECSENT	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", "MESSAGESSENT", "ALLSENT",
                "BYTESSENT", "BYTESPERSECSENT", "MESSAGESRECV", "BYTESRECV",
                "ALLRECV", "BYTESPERSECRECV", "AVGSENT", "AVGRECV",
                "LASTPINGTIME", "LASTPINGDURATION", "AVGPINGDURATION",
                "MAXPINGTIME", "MAXPINGDURATION"}
    file = io.open("res.txt", "w+")
    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 Рабочего места QUIK
icon_type	NUMBER	Тип отображаемой иконки в сообщении. Возможные значения и вид иконки: _ «1» (по умолчанию) – ; _ «2» – ; _ «3» – . 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 слеша («\»). 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())
End
```

2.1.8 sysdate

Функция возвращает системные дату и время с точностью до микросекунд.

Формат вызова:

TABLE os.sysdate()

Функция возвращает таблицу Lua с параметрами, описание которых приведено в п. 4.23.

> **Функция использует системные вызовы операционной системы Windows и ее значения зависят от интервала между прерываниями системного таймера.**

2.1.9 isDarkTheme

Функция позволяет получить информацию о том, в какой теме оформления интерфейса в настоящий момент работает терминал: стандартной или темной.

Формат вызова:

BOOLEAN isDarkTheme()

Функция возвращает «true», если используется темная тема, иначе – «false».

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.**

Торговые данные передаются в скрипт только в том случае, если параметры по данному инструменту 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)**

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	Таблица с параметрами фирмы

2.2.2 OnAllTrade

Функция вызывается терминалом QUIK при получении обезличенной сделки.

Формат вызова:

OnAllTrade(TABLE alltrade)

Параметры:

Параметр	Тип	Описание
alltrade	TABLE	Таблица с параметрами обезличенной сделки

2.2.3 OnTrade

Функция вызывается терминалом QUIK при получении сделки или при изменении параметров существующей сделки.

Формат вызова:

OnTrade(TABLE trade)

Параметры:

Параметр	Тип	Описание
trade	TABLE	Таблица с параметрами сделки

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	Таблица с параметрами заявки

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	Таблица «Позиции участника по торговым счетам»

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	Таблица «Ограничения по клиентским счетам»

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	Таблица с параметрами удаляемого лимита по срочному рынку

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	Таблица с описанием позиции по срочному рынку

2.2.9 OnMoneyLimit

Функция вызывается терминалом QUIK при получении изменений по денежной позиции клиента.

Формат вызова:

OnMoneyLimit(TABLE mlimit)

Параметры:

Параметр	Тип	Описание
mlimit	TABLE	Таблица «Позиции по деньгам»

2.2.10 OnMoneyLimitDelete

Функция вызывается терминалом QUIK при удалении денежной позиции.

Формат вызова:

OnMoneyLimitDelete(TABLE mlimit_del)

Параметры:

Параметр	Тип	Описание
mlimit_del	TABLE	Таблица с параметрами удаляемой позиции по денежным средствам

2.2.11 OnDepoLimit

Функция вызывается терминалом QUIK при получении изменений позиции по инструментам.

Формат вызова:

OnDepoLimit(TABLE dlimit)

Параметры:

Параметр	Тип	Описание
dlimit	TABLE	Таблица «Позиции по инструментам»

2.2.12 OnDepoLimitDelete

Функция вызывается терминалом QUIK при удалении позиции клиента по инструментам.

Формат вызова:

OnDepoLimitDelete(TABLE dlimit_del)

Параметры:

Параметр	Тип	Описание
dlimit_del	TABLE	Таблица с параметрами удаляемой позиции по инструментам

2.2.13 OnAccountPosition

Функция вызывается терминалом QUIK при изменении позиции участника по денежным средствам.

Формат вызова:

OnAccountPosition(TABLE acc_pos)

Параметры:

Параметр	Тип	Описание
acc_pos	TABLE	Таблица «Позиции участника по деньгам»

2.2.14 OnNegDeal

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

Формат вызова:

OnNegDeal(TABLE neg_deals)

Параметры:

Параметр	Тип	Описание
neg_deals	TABLE	Таблица с параметрами заявки на внебиржевые сделки

2.2.15 OnNegTrade

Функция вызывается терминалом QUIK при получении сделки для исполнения или при изменении параметров существующей сделки для исполнения.

Формат вызова:

OnNegTrade(TABLE neg_trade)

Параметры:

Параметр	Тип	Описание
neg_trade	TABLE	Таблица с параметрами сделки для исполнения

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	Таблица с параметрами стоп-заявки

2.2.17 OnTransReply

Функция OnTransReply вызывается терминалом QUIK при получении ответа на транзакцию пользователя, отправленную с помощью любого плагина Рабочего места QUIK (в том числе QLua). Для транзакций, отправленных с помощью Trans2quik.dll, QPILE или динамической загрузки транзакций из файла, функция не вызывается.

Формат вызова:

OnTransReply(TABLE trans_reply)

Параметры:

Параметр	Тип	Описание
trans_reply	TABLE	Таблица с описанием транзакций

> **Параллельно исполняемые скрипты получают оповещения об ответах на свои транзакции и на транзакции, выставленные другими скриптами.**

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 инструмента

> **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
			message("Спрос " .. tbid.param_image)
		end
	end
end
```

2.2.19 OnQuote

Функция вызывается терминалом QUIK при получении изменения стакана котировок.

OnQuote(STRING class_code, STRING sec_code)

Параметры:

Параметр	Тип	Описание
class_code	STRING	Код класса
sec_code	STRING	Код инструмента

> **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
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 в следующих случаях:

- смена сервера QUIK внутри торговой сессии;
- смена пользователя, которым выполняется подключение к серверу QUIK, внутри торговой сессии;
- смена сессии.

OnCleanUp()

При выполнении сразу нескольких из перечисленных условий, функция OnCleanUp() вызывается терминалом QUIK для каждого из них.

> **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 flag)

Функция возвращает количество миллисекунд, которое дается скрипту на завершение работы. Если функция не возвращает число, то таймаут завершения работы скрипта остается равным 5 секундам. Если скрипт не завершит работу за отведенное время, возвращенное в функции OnStop, или за 5 секунд, то инициируется режим экстренного завершения работы интерпретатора Lua. Если через 1 секунду срипт не завершил работу, поток Lua принудительно завершается функцией terminateThread().

> **При принудительной остановке могут быть сбои в работе терминала QUIK.**

При остановке или удалении работающего скрипта Lua из диалога управления «Доступные скрипты» параметр вызова flag принимает значение «1». При закрытии терминала QUIK – значение «2».

Пример:

```
function OnStop(flag)
       stopped = true
       return 3000 -- задается таймаут в 3 секунды
end
function OnStop(flag)
       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().**

2.2.26main

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.

main()

Пример:

```
function main()
       while true do
              message(os.date()) --раз в секунду выводит текущие дату и время
              sleep(1000)
       end
end
```

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».

TABLE getItem (STRING TableName, NUMBER Index)

В случае неуспешного завершения функция возвращает «nil».

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.

TABLE order NUMBER indx 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».

NUMBER 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])

> **При возникновении ошибки в функции fn функция SearchItems прерывает свою работу и возвращает «nil».**

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 одно из следующих значений:
  - true – текущий индекс учитывается в результате;
  - false – текущий индекс не учитывается в результате;
  - nil – поиск прерывается, функция SearchItems возвращает таблицу с индексами, найденными ранее, включая текущий индекс.
- 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» и «SearchItems»

TableName	Таблица
firms	Фирмы
classes	Классы
securities	Инструменты
trade_accounts	Торговые счета
client_codes	* Коды клиентов
all_trades	Обезличенные сделки
account_positions	Позиции участника по деньгам
orders	Заявки
futures_client_holding	Позиции по клиентским счетам (фьючерсы)
futures_client_limits	Ограничения по клиентским счетам
money_limits	Позиции по деньгам
depo_limits	Позиции по инструментам

TableName	Таблица
trades	Сделки
stop_orders	Стоп-заявки
neg_deals	Заявки на внебиржевые сделки
neg_trades	Сделки для исполнения
neg_deal_reports	Отчеты по сделкам для исполнения
firm_holding	Позиции участника по инструментам
account_balance	Позиции участника по торговым счетам
ccp_holdings	Обязательства и требования по активам
rm_holdings	Валюта: обязательства и требования по активам

* – 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 «,».

STRING getClassesList()

Пример:

```
list = getClassesList()
```

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

```
OPTEXP,USDRUB,PSOPT,PSFUT,SPBFUT,
```

3.2.2 getClassInfo

Функция предназначена для получения информации о классе.

TABLE getClassInfo (STRING)

Возвращает таблицу Lua, содержащую описание класса:

Параметр	Тип	Описание
firmid	STRING	Код фирмы
name	STRING	Наименование класса
code	STRING	Код класса
npars	NUMBER	Количество параметров в классе
nsecs	NUMBER	Количество инструментов в классе

3.2.3 getClassSecurities

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

STRING 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 позициям.

TABLE getMoney (STRING client_code, STRING firmid, STRING tag, STRING currcode)
Функция возвращает таблицу Lua с параметрами:

Параметр	Тип	Описание
money_open_limit	NUMBER	Входящий лимит по денежным средствам
money_limit_locked_nonmarginal_value	NUMBER	Стоимость немаржинальных инструментов в заявках на покупку
money_limit_locked	NUMBER	Заблокированное в заявках на покупку количество денежных средств
money_open_balance	NUMBER	Входящий остаток по денежным средствам
money_current_limit	NUMBER	Текущий лимит по денежным средствам
money_current_balance	NUMBER	Текущий остаток по денежным средствам
money_limit_available	NUMBER	Доступное количество денежных средств

3.3.2 getMoneyEx

Функция предназначена для получения информации по денежным позициям указанного типа.

TABLE getMoneyEx(STRING firmid, STRING client_code, STRING tag, STRING currcode, NUMBER limit_kind)
Функция возвращает таблицу Lua с параметрами таблицы «Позиции по деньгам».

В случае ошибки функция возвращает «nil».

3.4 Функция для получения позиций по инструментам

3.4.1 getDepo

Функция предназначена для получения позиций по инструментам.

TABLE getDepo (STRING client_code, STRING firmid, STRING sec_code, STRING trdaccid)

Функция возвращает таблицу Lua с параметрами:

Параметр	Тип	Описание
depo_limit_locked_buy_value	NUMBER	Стоимость инструментов, заблокированных на покупку
depo_current_balance	NUMBER	Текущий остаток по инструментам
depo_limit_locked_buy	NUMBER	Количество лотов инструментов, заблокированных на покупку
depo_limit_locked	NUMBER	Заблокированное Количество лотов инструментов
depo_limit_available	NUMBER	Доступное количество инструментов
depo_current_limit	NUMBER	Текущий лимит по инструментам
depo_open_balance	NUMBER	Входящий остаток по инструментам
depo_open_limit	NUMBER	Входящий лимит по инструментам

3.4.2 getDepoEx

Функция предназначена для получения позиций по инструментам указанного типа.

TABLE getDepoEx(STRING firmid, STRING client_code, STRING sec_code, STRING trdaccid, NUMBER limit_kind)

Функция возвращает таблицу Lua с параметрами таблицы «Позиции по инструментам».

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 firmid, STRING trdaccid, NUMBER limit_type, STRING currcode)

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

Функция возвращает таблицу Lua с параметрами таблицы «Ограничения по клиентским счетам».

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 firmid, STRING trdaccid, STRING sec_code, NUMBER type)

Функция возвращает таблицу Lua с параметрами Таблицы «Позиции по клиентским счетам».

В случае ошибки функция возвращает «nil».

3.7 Функция для получения информации по инструменту

3.7.1 getSecurityInfo

Функция предназначена для получения информации по инструменту.

TABLE 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.

TABLE 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 инструменту

3.9.1 getQuoteLevel2

Функция предназначена для получения стакана по указанному классу и инструменту.

TABLE getQuoteLevel2 (STRING class_code, STRING sec_code)

Функция возвращает таблицу Lua с параметрами:

> **При отсутствии и спроса, и предложения функция возвращает таблицу без параметров bid и offer.**

Параметр	Тип	Описание
bid_count	STRING	Количество котировок покупки, 6 знаков после запятой
offer_count	STRING	Количество котировок продажи, 6 знаков после запятой
bid	TABLE	Котировки спроса (покупки). При отсутствии спроса возвращается пустая строка
offer	TABLE	Котировки предложений (продажи). При отсутствии предложения возвращается пустая строка

Каждый элемент массивов «bid» и «offer» содержит параметры price и quantity:

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

Примеры

Нет спроса и предложения:

```
{'bid_count': '0.000000',
'offer_count': '0.000000'
}
```

Есть спрос и предложение:

```
{'bid_count': '3.000000',
'offer_count': '3.000000',
'bid': {{'price':'11.3', 'quantity':'10'},
        {'price':'11.4', 'quantity':'20'},
        {'price':'11.5', 'quantity':'30'},
       },
'offer': {
          {'price':'11.6', 'quantity':'10'},
          {'price':'11.7', 'quantity':'15'},
          {'price':'11.8', 'quantity':'18'}
       },
}
```

Есть спрос, но нет предложения:

```
{'bid_count': '3.000000',
'offer_count': '0.000000',
'bid': {{'price':'11.3', 'quantity':'10'},
        {'price':'11.4', 'quantity':'20'},
        {'price':'11.5', 'quantity':'30'},
       },
'offer': ''
}
```

Есть предложение, но нет спроса:

```
{'bid_count': '0.000000',
'offer_count': '3.000000',
'bid': '',
'offer': {
          {'price':'11.6', 'quantity':'10'},
          {'price':'11.7', 'quantity':'15'},
          {'price':'11.8', 'quantity':'18'}
       }
}
```

3.10 Функции для работы с графиками

3.10.1getLinesCount

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

NUMBER getLinesCount (STRING tag)

Возвращает число – количество линий на графике.

3.10.2getNumCandles

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

NUMBER getNumCandles (STRING tag)

Возвращает число – количество свечек по выбранному идентификатору.

3.10.3getCandlesByIndex

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.

3.10.4CreateDataSource

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 инструмента,
- **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, код инструмента, интервал или параметр, то возвращается «nil». При этом **error_desc** содержит описание ошибки.

> **Функцию CreateDataSource можно использовать только внутри функций main() и callback.**

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

Параметр	Значение интервала
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
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
SetEmptyCallback	function	Позволяет получать данные с сервера без указания функции обратного вызова

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\t%.4f %02d.%02d.%04d
%02d:%02d:%02d.%03d",
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.

3.11.1sendTransaction

Функция предназначена для отправки транзакций в торговую систему.

Формат вызова:

STRING result sendTransaction(TABLE transaction)

Параметры:

- **result** – строка, содержащая текст ошибки, если она случилась при обработке транзакции.
- **transaction** – таблица с параметрами транзакции

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»/ «Формат .tri-файла с параметрами транзакций»).

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

Пример заполнения полей таблицы **transaction**:

```
transaction = {
ACCOUNT="YY0070001234",
CLIENT_CODE="XXX",
TYPE="M",
TRANS_ID="7",
CLASSCODE="TQBR",
SECCODE="HYDR",
ACTION="NEW_ORDER",
OPERATION="B",
PRICE="0",
QUANTITY="15"
}
```

3.11.2CalcBuySell

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 инструмента;
- **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 funstion!!")
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 «Текущие торги»

3.12.1getParamEx

Функция предназначена для получения значений всех параметров биржевой информации из таблицы «Текущие торги». С помощью этой функции можно получить любое из значений Таблицы текущих торгов для заданных кодов класса и инструмента, подробнее см. п. 3.12.3.

Формат вызова:

TABLE 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 «Текущие торги». 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.
result	STRING	Результат выполнения операции. Возможные значения: _ «0» – ошибка; _ «1» – параметр найден

3.12.2getParamEx2

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, подробнее см. п. 3.12.3. Для отказа от получения какого-либо параметра воспользуйтесь функцией CancelParamRequest.

Формат вызова:

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

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента;
- **param_name** – код параметра.

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

3.12.3Особенности получения значений Таблицы текущих торгов

Получение значений параметров Таблицы текущих торгов выполняется функцией getParamEx() или getParamEx2(). Актуальные значения параметров можно получить, если выполнен заказ данных. Способы выполнения заказа данных зависят от режимов, которые задаются в основных настройках РМ QUIK (пункт меню **Система / Настройки / Основные настройки...**, раздел «Программа» / «Получение данных» / «Котировки»):

- Вручную, ключив режим «по выбранным классам» и указав необходимые параметры и инструменты на классе;
- При включенном режиме «"умный" заказ данных» и открытой Таблице текущих торгов с необходимыми параметрами и инструментами на классе;
- Автоматически из скрипта Lua с помощью функций ParamRequest или CreateDataSource, при включенном режиме «"умный" заказ данных».

> **Терминал QUIK автоматически заказывает параметры, необходимые для корректного расчета позиций при включенном режиме Рабочего места QUIK «"умный" заказ данных»).**

Получение служебных идентификаторов параметров

Набор параметров, отображаемых в таблице «Текущие торги» Рабочего места QUIK, может отличаться в зависимости от перечня торговых систем и режимов торгов, к которым предоставляется доступ данным сервером системы QUIK.

Для обращения к данным параметрам необходимо узнать их имена (служебные идентификаторы), которые могут быть получены следующим образом:

- В Рабочем месте QUIK создайте таблицу «Текущие торги» с необходимым набором параметров. В таблице достаточно одной строки с инструментом, поскольку требуется получить только список заголовков. В окне настройки таблицы включите флажок «Транспонировать».
- Запустите MS Excel.
- В контекстном меню таблицы «Текущие торги» выберите пункт «Вывод через DDE-сервер». В открывшемся диалоге укажите имя рабочей книги MS Excel (как правило, это «Книга1») и листа («Лист1»). Также необходимо заполнить значениями следующие поля:
  - «Ряд» – «1»;
  - «Колонка» – «1»;
  - Флажок «С заголовками строк» должен быть включен;
  - Флажок «С заголовками столбцов» должен быть включен;
  - Флажок «Формальные заголовки» должен быть выключен.
- Выведите данные в таблицу MS Excel нажатием кнопки «Вывести сейчас». Таблица должна заполниться данными, при этом в первой колонке указываются заголовки в том виде, как они показываются в Рабочем месте QUIK.
- Вернитесь в диалог настройки экспорта в MS Excel в Рабочем месте QUIK и измените следующие параметры:

- «Колонка» – «2»;
- Флажок «Формальные заголовки» должен быть включен;

- Снова выведите данные в MS Excel нажатием кнопки «Вывести сейчас». Первая колонка содержит заголовки параметров, а вторая – их служебные идентификаторы, которые можно использовать в качестве аргумента param_name функции getParamEx() или getParamEx2(), например:

	1	2	3
1			MSNG
2	Номинал	SEC_FACE_VALUE	1
3	Валюта	SEC_FACE_UNIT	SUR
4	Точность	SEC_SCALE	3
5	Шаг цены	SEC_PRICE_STEP	0,001

Если нет MS Excel, то добавление параметров можно выполнить альтернативным способом в любом текстовом редакторе:

- В Рабочем месте QUIK создайте таблицу «Текущие торги» с необходимым набором параметров. В таблице достаточно одной строки с инструментом, поскольку требуется получить только список заголовков. В окне настройки таблицы включите флажок «Транспонировать».
- В пункте меню **Система/Настройки, Основные настройки…**, раздел «Программа»/ «Буфер обмена» включите признаки «Выводить заголовки строк», «Выводить заголовки столбоцов», «Формальное представление данных» и «Формальное представление заголовков строк и столбцов».
- Скопируйте все даннные из Таблицы текущих торгов (пункт контекстного меню таблицы «Копировать все»).
- Откройте любой текстовый редактор и вставьте скопированные данные.

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», коду клиента «client_code» и сроку расчетов «limit_kind» со значением «0».

Формат вызова:

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

Функция возвращает таблицу Lua с параметрами:

№	Параметр	Тип	Описание	Краткое наименование
1	is_leverage	STRING	Признак использования схемы кредитования с контролем текущей стоимости активов. Возможные значения: _ «МЛ» – используется схема ведения позиции «по плечу», «плечо» рассчитано по значению Входящего лимита, _ «МП» – используется схема ведения позиции «по плечу», «плечо» указано явным образом, _ «МОП» – используется схема ведения позиции «лимит на открытую позицию»; _ «МД» – используется схема ведения позиции «по дисконтам»; _ «С» – используется схема ведения позиций «срочный рынок». Для клиентов срочного рынка без единой денежной позиции; _ <пусто> – используется схема ведения позиции «по лимитам»	Тип клиента
2	in_assets	STRING	Оценка собственных средств клиента до начала торгов	Вход. активы
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 инструментам, принимаемым в обеспечение	Лонги МО

№	Параметр	Тип	Описание	Краткое наименование
8	val_long_asset	STRING	Оценка стоимости длинных позиций по немаржинальным инструментам, 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 инструментов, принимаемых в обеспечение	Блок. пок. маржин.
16	locked_buy_asset	STRING	Оценка стоимости активов в заявках на покупку немаржинальных инструментов, принимаемых в обеспечение	Блок.пок. обесп.
17	locked_sell	STRING	Оценка стоимости активов в заявках на продажу маржинальных инструментов	Блок. продажа
18	locked_value_coef	STRING	Оценка стоимости активов в заявках на покупку немаржинальных инструментов	Блок. пок. немарж.
19	in_all_assets	STRING	Оценка стоимости всех позиций клиента в ценах закрытия предыдущей торговой сессии, включая позиции по немаржинальным инструментам	ВходСредства
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

№	Параметр	Тип	Описание	Краткое наименование
23	lim_buy	STRING	Оценка денежных средств, доступных для покупки маржинальных инструментов	На покупку
24	lim_sell	STRING	Оценка стоимости маржинальных инструментов, доступных для продажи	На продажу
25	lim_non_margin	STRING	Оценка денежных средств, доступных для покупки немаржинальных инструментов	НаПокупНеМаржин
26	lim_buy_asset	STRING	Оценка денежных средств, доступных для покупки инструментов, 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 инструментам обеспечения) и коротких позиций по инструментам, дисконтов корреляции между инструментами, а также дисконтов на задолженности по валютам, не покрытые обеспечением по инструментам в этих же валютах	Сумма дисконтов

№	Параметр	Тип	Описание	Краткое наименование
32	assets_without_hc	STRING	Суммарная величина денежных остатков, стоимости длинных позиций по инструментам обеспечения и стоимости коротких позиций, без учета дисконтирующих коэффициентов, без учета неттинга стоимости инструментов в рамках объединенной позиции по инструментам и без учета корреляции между инструментами	ТекАктБезДиск
33	status_coef	STRING	Отношение суммы дисконтов к текущим активам без учета дисконтов	Статус счета
34	*varmargin	STRING	Текущая вариационная маржа по позициям клиента, по всем инструментам	Вариац. 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 срочном рынке Московской Биржи, 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>-<Код позиции>». Пример: «SUR-EQTV»	Парам. расч.
41	init_margin	STRING	Значение начальной маржи. Заполняется для клиентов типа «МД»	Нач.маржа
42	init_margin_net	STRING	Значение начальной маржи по принципу «нетто». Заполняется для клиентов типа «МД+»	Нач.маржа нетто

№	Параметр	Тип	Описание	Краткое наименование
43	min_margin	STRING	Значение минимальной маржи. Заполняется для клиентов типа «МД»	Мин.маржа
44	min_margin_net	STRING	Значение минимальной маржи по принципу «нетто». Заполняется для клиентов типа «МД+»	Мин.маржа нетто
45	corrected_margin	STRING	Значение скорректированной маржи. Заполняется для клиентов типа «МД»	Скор.маржа
46	client_type	STRING	Тип клиента	Тип клиента
47	portfolio_value	STRING	Стоимость портфеля. Для клиентов типа «МД» возвращается значение для строк с максимальным сроком расчетов limit_kind	Стоимость портфеля
48	*start_limit_open_pos	STRING	Лимит открытых позиций на начало дня	ЛимОткрПозНачДня
49	*total_limit_open_pos	STRING	Лимит открытых позиций	ЛимОткрПоз
50	*limit_open_pos	STRING	Планируемые чистые позиции	ПланЧистПоз
51	*used_lim_open_pos	STRING	Текущие чистые позиции	ТекЧистПоз
52	*acc_var_margin	STRING	Накопленная вариационная маржа	НакопВарМаржа
53	*cl_var_margin	STRING	Вариационная маржа по итогам промклиринга	ВарМаржаПромклир.
54	*opt_liquid_cost	STRING	Ликвидационная стоимость опционов	ЛиквСтоимОпционов
55	*fut_asset	STRING	Сумма оценки средств клиента на срочном рынке	СумАктивовНаСрчРынке
56	*fut_total_asset	STRING	Сумма оценки собственных средств клиента на фондовом и срочном рынках	ПолнСтоимостьПортфеля
57	*fut_debt	STRING	Текущая задолженность на срочном рынке	ТекЗадолжНаСрчРынке
58	*fut_rate_asset	STRING	Достаточность средств	Дост. Средств
59	*fut_rate_asset_open	STRING	Достаточность средств (под открытые позиции)	Дост. Средств (ОткрПоз)
60	*fut_rate_go	STRING	Коэффициент ликвидности ГО	КоэффЛикв ГО
61	*planed_rate_go	STRING	Ожидаемый коэффициент ликвидности ГО	Ожид. КоэффЛикв ГО

№	Параметр	Тип	Описание	Краткое наименование
62	*cash_leverage	STRING	Cash Leverage	Cash Leverage
63	*fut_position_type	STRING	Тип позиции на срочном рынке. Возможные значения: _ «0» – нет позиции; _ «1» – фьючерсы; _ «2» – опционы; _ «3» – фьючерсы и опционы	ТипПозНаСрчРынке
64	*fut_accured_int	STRING	Накопленный доход с учетом премии по опционам и биржевым сборам	НакопДоход
65	rcv1	STRING	Норматив покрытия риска 1. Рассчитывается как разница параметров «Стоимость портфеля» и «Начальная маржа». Для клиентов типа «МД» и «МД+»	НПР1
66	rcv1_net	STRING	Норматив покрытия риска 1. Рассчитывается как разница параметров «Стоимость портфеля» и «Начальная маржа нетто». Для клиентов типа «МД+»	НПР1 нетто
67	rcv2	STRING	Норматив покрытия риска 2. Рассчитывается как разница параметров «Стоимость портфеля» и «Минимальная маржа». Для клиентов типа «МД» и «МД+»	НПР2
68	rcv2_net	STRING	Норматив покрытия риска 2. Рассчитывается как разница параметров «Стоимость портфеля» и «Минимальная маржа нетто». Для клиентов типа «МД+»	НПР2 нетто
69	demand	STRING	Требование, если стоимость портфеля меньше стоимости начальной маржи, то считается как разность начальной маржи и портфеля, иначе «0». Заполняется для клиентов типа «МД»	Требование
70	current_bal	STRING	Сумма текущих остатков с денежных лимитов клиента	
71	open_pos	STRING	Оценка открытых позиций клиента с учетом активных заявок. Заполняется для клиентов типа «МП»	
72	money_locked	STRING	Сумма блокировок по активным заявкам с денежных лимитов	

№	Параметр	Тип	Описание	Краткое наименование
73	is_marginal	STRING	«1» для клиентов типа «МЛ» (тип кредитования «по лимитам»)	
74	fundslevel	STRING	Уровень достаточности средств (УДС). Рассчитывается только для клиентов типа «МД» и «МД+», в остальных случаях принимает значение «9.99»	УДС

* – параметр заполняется для клиентов с настроенной единой денежной позицией и для клиентов срочного рынка без единой денежной позиции.

** – 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 сроку расчетов «limit_kind» со значением, заданным пользователем. Необязательные аргументы код позиции «board_tag» и валюта «currency» задают параметры расчета таблицы «Клиентский портфель», при их отсутствии используются код позиции и валюта по умолчанию для указанного участника торгов («firmid»). При этом в таблице с результатом в поле «curr_tag» заполняются параметры, для которых был выполнен расчет в формате «<Валюта>-<Код позиции>».

Формат вызова:

```
TABLE getPortfolioInfoEx (STRING firm_id, STRING client_code, NUMBER limit_kind,
[STRING board_tag, STRING currency])
```

> **Для получения значений параметров таблицы «Клиентский портфель» для клиентов срочного рынка без единой денежной позиции необходимо указать в качестве «client_code» – торговый счет на срочном рынке, а в качестве «limit_kind» – 0.**

Функция возвращает таблицу Lua с параметрами таблицы «Клиентский портфель». Описание параметров см. в п. 3.13.1.

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: _ «0» – не маржинальная; _ «1» – маржинальная
2	is_asset_sec	STRING	Принадлежность инструмента к списку инструментов, принимаемых в обеспечение. Возможные значения: _ «0» – не принимается в обеспечение; _ «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

№	Параметр	Тип	Описание
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 инструментов и их рыночной оценки
19	spread_hc	STRING	Коэффициент корреляции между инструментами
20	can_buy_own	STRING	Максимально возможное количество инструментов в заявке на покупку этого инструмента на этом классе на собственные средства клиента, исходя из цены лучшего предложения
21	can_sell_own	STRING	Максимально возможное количество инструментов 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	Срок расчетов. Возможные значения: положительные целые числа, начиная с «0», соответствующие срокам расчетов из таблицы «Позиции по инструментам»: «0» – T0, «1» – T1, «2» – T2 и т.д.
2	d_long	STRING	Эффективный начальный дисконт для длинной позиции. Заполняется для клиентов типа «МД»
3	d_min_long	STRING	Эффективный минимальный дисконт для длинной позиции. Заполняется для клиентов типа «МД»
4	d_short	STRING	Эффективный начальный дисконт для короткой позиции. Заполняется для клиентов типа «МД»
5	d_min_short	STRING	Эффективный минимальный дисконт для короткой позиции. Заполняется для клиентов типа «МД»
6	client_type	STRING	Тип клиента. Возможные значения: _ «1» – «МЛ»; _ «2» – «МП»; _ «3» – «МОП»; _ «4» – «МД»; _ «5» - «МД+»
7	is_long_allowed	STRING	Признак того, является ли инструмент разрешенным для покупки на заемные средства. Возможные значения: «1» – разрешен, «0» – не разрешен. Заполняется для клиентов типа «МД»
8	is_short_allowed	STRING	Признак того, является ли инструмент разрешенным для продажи на заемные средства. Возможные значения: «1» – разрешен, «0» – не разрешен. 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, поддержаны 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 заголовок 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:

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

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

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)
```

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

3.15.6 DestroyTable

Функция закрывает окно таблицы с идентификатором «t_id».

Формат вызова:

```
BOOLEAN 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».

> **Добавление строк возможно только в созданное окно таблицы, после вызова функции CreateWindow (описание см. в п. 3.15.4).**

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.

3.15.8IsWindowClosed

Функция предназначена для определения состояния окна таблицы «t_id».

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

Формат вызова:

```
BOOLEAN IsWindowClosed(NUMBER t_id)
```

Функция возвращает:

- «false» – если окно таблицы «t_id» открыто в терминале;
- «true» – если окно таблицы «t_id» закрыто пользователем вручную. Окно может быть открыто повторно с помощью функции CreateWindow;
- «nil» – если таблица «t_id» не существует (указан несуществующий / некорректный идентификатор или таблица удалена функцией DestroyTable).

3.15.9GetCell

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 значение 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:

BOOLEAN 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.

> **ВАЖНО! Недопустим вызов функций Clear и DestroyTable для t_id внутри функции обратного вызова f_cb, назначенной на таблицу с данным t_id.**

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_MBUTTONDBLCLK – 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
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.

Формат вызова:

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.

3.16.1AddLabel

Добавляет метку с заданными параметрами.

Формат вызова:

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). Используются картинки формата *.bmp, *.jpeg

№	Параметр	Тип	Описание
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]
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 фона картинки. 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

3.16.2DelLabel

Удаляет метку с заданными параметрами.

Формат вызова:

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».

3.16.3DelAllLabels

Команда удаляет все метки на диаграмме с указанным графиком.

Формат вызова:

BOOLEAN DelAllLabels(STRING chart_tag)

Параметры:

- **chart_tag** – тег графика, к которому привязывается метка.

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

3.16.4GetLabelParams

Команда позволяет получить параметры метки.

Формат вызова:

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.**

3.16.5SetLabelParams

Функция задает параметры для метки с указанным идентификатором.

Формат вызова:

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 инструменту.

Формат вызова:

BOOLEAN Subscribe_Level_II_Quotes(STRING class_code, STRING sec_code)

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента.

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 инструменту.

Формат вызова:

BOOLEAN Unsubscribe_Level_II_Quotes(STRING class_code, STRING sec_code)

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента.

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 инструменту.

Формат вызова:

BOOLEAN IsSubscribed_Level_II_Quotes (STRING class_code, STRING sec_code)

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента.

Функция возвращает «true», если стакан по классу **class_code** и инструменту **sec_code** уже заказан.

3.18 Функции для заказа параметров Таблицы текущих торгов

3.18.1 ParamRequest

Функция заказывает получение параметров Таблицы текущих торгов.

Формат вызова:

BOOLEAN ParamRequest(STRING class_code, STRING sec_code, STRING db_name)

> **Для корректной работы функции включите в настройках Рабочего места QUIK режим «"умный" заказ данных» (меню Система / Настройки / Основные настройки..., раздел «Программа» / «Получение данных» / «Котировки»).**

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента;
- **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.

Формат вызова:

BOOLEAN CancelParamRequest(STRING class_code, STRING sec_code, STRING db_name)

> **Для корректной работы функции включите в настройках Рабочего места QUIK режим «"умный" заказ данных» (меню Система / Настройки / Основные настройки..., раздел «Программа» / «Получение данных» / «Котировки»).**

Параметры:

- **class_code** – код класса;
- **sec_code** – код инструмента;
- **db_name** – код параметра.

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

> **При использовании функций ParamRequest() и CancelParamRequest() отказ от заказанных параметров Таблицы текущих торгов возможен, если в скрипте используется функция getParamEx2().**

> **Если будет использоваться функция getParamEx(), то функция CancelParamRequest() не выполнит отмену получения параметров Таблицы текущих торгов.**

3.19 Функции для получения информации по единой денежной позиции

3.19.1 getTrdAccByClientCode

Функция возвращает торговый счет срочного рынка, соответствующий коду клиента фондового рынка с единой денежной позицией.

Формат вызова:

```
STRING getTrdAccByClientCode(STRING firmid, STRING client_code)
```

Параметры:

- **firmid** – идентификатор фирмы фондового рынка;
- **client_code** – код клиента.

Функция возвращает строку с торговым счетом срочного рынка, если указанный код клиента фондового рынка имеет единую денежную позицию, иначе – «nil».

3.19.2 getTrdAccByClientCodeEx

Функция возвращает таблицу с торговыми счетами, соответствующими коду клиента с единой денежной позицией.

Формат вызова:

```
TABLE getTrdAccByClientCodeEx(STRING firmid, STRING client_code)
```

Параметры:

- **firmid** – идентификатор фирмы фондового рынка;
- **client** – кода клиента.

Функция возвращает таблицу, содержащую все торговые счета, соответствующие коду клиента, если указанный код клиента фондового рынка имеет единую денежную позицию. Если данные отсутствуют на сервере, возвращается пустая таблица. Если данные не удалось получить из-за ошибки, вместо таблицы функция возвращает «nil».

3.19.3 getClientCodeByTrdAcc

Функция возвращает код клиента фондового рынка с единой денежной позицией, соответствующий торговому счету срочного рынка.

Формат вызова:

```
STRING getClientCodeByTrdAcc(STRING firmid, STRING trdaccid)
```

Параметры:

- **firmid** – идентификатор фирмы фондового рынка;
- **trdaccid** – торговый счет срочного рынка.

Функция возвращает строку с кодом клиента, если указанный торговый счет имеет единую денежную позицию, иначе – «nil».

3.19.4isUcpClient

Функция предназначена для получения признака, указывающего имеет ли клиент единую денежную позицию.

Формат вызова:

```
BOOLEAN isUcpClient(STRING firmid, STRING client)
```

Параметры:

- **firmid** – идентификатор фирмы фондового рынка;
- **client** – кода клиента фондового рынка или торговый счет срочного рынка.

Функция возвращает «true», если указанный клиент имеет единую денежную позицию, 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 инструментов 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	Набор битовых флагов
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

Параметр	Тип	Описание
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
datetime	TABLE	Дата и время
period	NUMBER	Период торговой сессии. Возможные значения: _ «0» – Открытие; _ «1» – Нормальный; _ «2» – Закрытие
open_interest	NUMBER	Открытый интерес
exchange_code	STRING	Код биржи в торговой системе
exec_market	STRING	Площадка исполнения
benchmark	STRING	Идентификатор индикативной ставки
trading_session_date	NUMBER	Дата биржевой торговой сессии
z_spread	NUMBER	Z-спред по цене

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
canceled_uid	NUMBER	Идентификатор пользователя, отказавшегося от сделки
account	STRING	Торговый счет
price	NUMBER	Цена

Параметр	Тип	Описание
qty	NUMBER	Количество инструментов 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	Набор битовых флагов
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
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 инструмента заявки
class_code	STRING	Код класса

Параметр	Тип	Описание
datetime	TABLE	Дата и время
bank_acc_id	STRING	Идентификатор расчетного счета/кода в клиринговой организации
broker_comission	NUMBER	Комиссия брокера. Отображается с точностью до 2 знаков
linked_trade	NUMBER	Номер витринной сделки в Торговой Системе для сделок РЕПО с ЦК и SWAP
period	NUMBER	Период торговой сессии. Возможные значения: _ «0» – Открытие; _ «1» – Нормальный; _ «2» – Закрытие
trans_id	NUMBER	Идентификатор транзакции. При превышении значения «9 223 372 036 854 775 807» поле принимает значение «-1», в этом случае рекомендуется использовать поле «trans_id_str»
trans_id_str	STRING	Идентификатор транзакции
kind	NUMBER	Тип сделки. Возможные значения: _ «1» – Обычная; _ «2» – Адресная; _ «3» – Первичное размещение; _ «4» – Перевод денег/инструментов; _ «5» – Адресная сделка первой части РЕПО; _ «6» – Расчетная по операции своп; _ «7» – Расчетная по внебиржевой операции своп; _ «8» – Расчетная сделка бивалютной корзины; _ «9» – Расчетная внебиржевая сделка бивалютной корзины; _ «10» – Сделка по операции РЕПО с ЦК; _ «11» – Первая часть сделки по операции РЕПО с ЦК; _ «12» – Вторая часть сделки по операции РЕПО с ЦК; _ «13» – Адресная сделка по операции РЕПО с ЦК; _ «14» – Первая часть адресной сделки по операции РЕПО с ЦК; _ «15» – Вторая часть адресной сделки по операции РЕПО с ЦК;

Параметр	Тип	Описание
		_ «16» – Техническая сделка по возврату активов РЕПО с ЦК; _ «17» – Сделка по спреду между фьючерсами разных сроков на один актив; _ «18» – Техническая сделка первой части от спреда между фьючерсами; _ «19» – Техническая сделка второй части от спреда между фьючерсами; _ «20» – Адресная сделка первой части РЕПО с корзиной; _ «21» – Адресная сделка второй части РЕПО с корзиной; _ «22» – Перенос позиций срочного рынка; _ «23» – Late correction-XLON; _ «24» – Not to mark-XLON; _ «25» – Previous Day Contra; _ «26» – Ordinary trade immediate publication – XLON; _ «27» – Inter Fund Transfer delayed publication – XOFF; _ «28» – Negotiated trade delayed publication – XLON; _ «29» – Negotiated Trade immediate publication – XLON; _ «30» – OTC Late Correction – XOFF; _ «31» – Ordinary Trade delayed publication – XLON; _ «32» – Ordinary Trade Immediate publication – XOFF; _ «33» – SI Late Correction; _ «34» – SI Trade immediate publication; _ «35» – SI Trade delayed publication; _ «36» – OTC trade delayed publication – XOFF; _ «37» – OTC MTF TBA 1; _ «38» – OTC trade – delayed publication – XOFF; _ «39» – Inter fund cross – delayed publication requested MTF TBA 1; _ «40» – Cancellation of OTC trade after date of publication MTF TBA 2; _ «41» – OTC MTF TBA 1; _ «42» – OTC Trade – delayed publication MTF TBA 2;

Параметр	Тип	Описание
		_ «43» – Inter fund cross – delayed publication requested MTF TBA 2; _ «44» – Cancellation of OTC trade after date of publication MTF TBA 2; _ «45» – OTC MTF TBA 3; _ «46» – OTC trade – delayed publication MTF TBA 3; _ «47» – Inter fund cross – delayed publication requested MTF TBA 3; _ «48» – Cancellation of OTC trade after date of publication MTF TBA 3; _ «49» – OTC MTF TBA 4; _ «50» – OTC trade – delayed publication MTF TBA 4; _ «51» – Inter fund cross – delayed publication requested MTF TBA 4; _ «52» – Cancellation of OTC trade after date of publication MTF TBA 4; _ «53» – Delayed Publication Late Correction XLON; _ «54» – No to Mark Late Correction XLON; _ «55» – Витринная сделка по операции своп; _ «58» – Витринная сделка по адресной операции воп; _ «59» – Витринная сделка по Беспоставочному свопу; _ «60» – Сделка FX SPOT; _ «61» – Сделка по Беспоставочному валютному форвардному контракту; _ «62» – Сделка FX DEPOSIT; _ «63» – Сделка по Валютному расчетному форвардному контракту; _ «67» – Обратная EES-сделка; _ «68» – Прямая обеспеченаая EES-сделка; _ «69» – Прямая необеспеченная EES-сделка; _ «70» – Прямая смешанная EES-сделка; _ «71» – Исполнение фьючерсов; _ «72» – Исполнение опциона; _ «73» – Истечение опциона; _ «74» – Техническая (предварительная) сделка с провайдером ликвидности;

Параметр	Тип	Описание
		_ «75» – Сделка вследствие добровольного выхода из вечного фьючерса (на основании поданной заявки); _ «76» – Сделка вследствие принудительного выхода из вечного фьючерса (реализация неудовлетворенного спроса); _ «77» – Сделка в связнном инструменте вследствие выхода из вечного фьючерса
clearing_bank_accid	STRING	Идентификатор счета в НКЦ (расчетный код)
canceled_datetime	TABLE	Дата и время 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
lseccode	STRING	Приоритетное обеспечение
*order_revision_number	NUMBER	Номер ревизии заявки, по которой была совершена сделка
*order_qty	NUMBER	Количество в заявке на момент совершения сделки, в лотах
*order_price	NUMBER	Цена в заявке на момент совершения сделки
order_exchange_code	STRING	Биржевой номер заявки
exec_market	STRING	Площадка исполнения
liquidity_indicator	NUMBER	Индикатор ликвидности. Возможные значения: _ «0» – не определено; _ «1» – по заявке мейкера; _ «2» – по заявке тейкера; _ «3» – вывод ликвидности; _ «4» – по заявке в период аукциона
extref	STRING	Внешняя ссылка, используется для обратной связи с внешними системами
ext_trade_flags	NUMBER	Расширенные флаги, полученные от шлюза напрямую, без вмешательства сервера QUIK. Поле не заполняется

Параметр	Тип	Описание
on_behalf_of_uid	NUMBER	UID пользователя, от имени которого совершена сделка
client_qualifier	NUMBER	Квалификатор клиента, от имени которого совершена сделка. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «3» – Legal Entity
client_short_code	NUMBER	Краткий идентификатор клиента, от имени которого совершена сделка
investment_decision_maker_qualifier	NUMBER	Квалификатор принявшего решение о совершении сделки. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
investment_decision_maker_short_code	NUMBER	Краткий идентификатор принявшего решение о совершении сделки
executing_trader_qualifier	NUMBER	Квалификатор трейдера, исполнившего заявку, по которой совершена сделка. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
executing_trader_short_code	NUMBER	Краткий идентификатор трейдера, исполнившего заявку, по которой совершена сделка
waiver_flag	NUMBER	Признак того, что транзакция совершена по правилам пре-трейда. Возможные значения битовых флагов: _ бит 0 (0x1) – RFPT; _ бит 1 (0x2) – NLIQ; _ бит 2 (0x4) – OILQ; _ бит 3 (0x8) – PRC; _ бит 4 (0x10)– SIZE; _ бит 5 (0x20) – ILQD
mleg_base_sid	NUMBER	Идентификатор базового инструмента на сервере для multileg-инструментов
side_qualifier	NUMBER	Квалификатор операции. Возможные значения: _ «0» – не определено; _ «1» – Buy; _ «2» – Sell; _ «3» – Sell short; _ «4» – Sell short exempt; _ «5» – Sell undiclosed

Параметр	Тип	Описание
otc_post_trade_indicator	NUMBER	OTC post-trade индикатор. Возможные значения битовых флагов: _ бит 0 (0x1) – Benchmark; _ бит 1 (0x2) – Agency cross; _ бит 2 (0x4) – Large in scale; _ бит 3 (0x8) – Illiquid instrument; _ бит 4 (0x10) – Above specified size; _ бит 5 (0x20) – Cancellations; _ бит 6 (0x40) – Amendments; _ бит 7 (0x80) – Special dividend; _ бит 8 (0x100) – Price improvement; _ бит 9 (0x200) – Duplicative; _ бит 10 (0x400) – Not contributing to the price discovery process; _ бит 11 (0x800) – Package; _ бит 12 (0x1000) – Exchange for Physical
capacity	NUMBER	Роль в исполнении заявки. Возможные значения: _ «0» –не определено; _ «1» – Agent; _ «2» – Principal; _ «3» – Riskless principal; _ «4» – CFG give up; _ «5» – Cross as agent; _ «6» – Matched Principal; _ «7» – Proprietary; _ «8» – Individual; _ «9» – Agent for other member; _ «10» – Mixed; _ «11» – Market maker
cross_rate	NUMBER	Кросс-курс валюты цены сделки к валюте расчетов по сделке
fixing_date	NUMBER	Дата фиксации котировки для расчетов. Заполняется для контрактов типа «NDF» (см. operation_type)
start_date	NUMBER	Дата, начиная с которой допускается совершение валютирования. Заполняется только для контрактов типа «FLEX FORWARD» (см. operation_type)

Параметр	Тип	Описание
operation_type	NUMBER	Тип совершаемой операции. Возможные значения: _ «-1» – «NOT_DEFINED»; _ «0» – «SPOT»; _ «1» – «FORWARD»; _ «2» – «SWAP»; _ «6» – «NDF»; _ «7» – «FLEX FORWARD»
spot_rate	NUMBER	Цена котировки по спот-инструменту в момент совершения сделки
ts_commission_currency	STRING	Код валюты комиссии торговой системы
broker_commission_currency	STRING	Код валюты комиссии брокера
trading_session	NUMBER	Идентификатор торговой сессии. Возможные значения: _ «0» –«Не определено»; _ «1» –«Основная сессия»; _ «2» –«Дополнительная сессия»; _ «3» –«Итоги дня»
benchmark	STRING	Идентификатор индикативной ставки
deposit_intent	NUMBER	Тип сделки депозита. Возможные значения: _ «0» – сделка не типа «депозит»; _ «1» – «Намерение»; _ «2» – «Депозит»; _ «3» – «Изъятие»; _ «4» – «Закрытие планового пополнения»; _ «5» – «Принудительное закрытие»; _ «6» – «Перенос пополнения»
open_repo2date	NUMBER	Расчетная дата исполнения РЕПО с открытой датой
open_repo2value	NUMBER	Стоимость выкупа РЕПО с открытой датой
trading_session_date	NUMBER	Дата биржевой торговой сессии
open_repo_evergreen_term	NUMBER	Период до исполнения второй части сделки РЕПО с открытой датой
z_spread	NUMBER	Z-спред по цене
percent_period	NUMBER	Период процентных выплат

* – параметр используется только для сделок, совершенных по заявкам, к которым применена транзакция замены заявки с сохранением номера.

4.5 Заявки

Описание параметров Таблицы заявок:

Параметр	Тип	Описание
order_num	NUMBER	Номер заявки в торговой системе
* flags	NUMBER	Набор битовых флагов
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. При превышении значения «9 223 372 036 854 775 807» поле принимает значение «-1», в этом случае рекомендуется использовать поле «trans_id_str»
trans_id_str	STRING	Идентификатор транзакции
client_code	STRING	Код клиента
price2	NUMBER	Цена выкупа
settlecode	STRING	Код расчетов
uid	NUMBER	Идентификатор пользователя
canceled_uid	NUMBER	Идентификатор пользователя, снявшего заявку
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 инструмента заявки

Параметр	Тип	Описание
class_code	STRING	Код класса заявки
datetime	TABLE	Дата и время
withdraw_datetime	TABLE	Дата и время снятия заявки
bank_acc_id	STRING	Идентификатор расчетного счета/кода в клиринговой организации
value_entry_type	NUMBER	Способ указания объема заявки. Возможные значения: _ «0» – по количеству, _ «1» – по объему
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

Параметр	Тип	Описание
exec_type	NUMBER	Тип исполнения заявки. Возможные значения: _ «0» – «Значение не указано»; _ «1» – «Немедленно или отклонить»; _ «2» – «Поставить в очередь»; _ «3» – «Снять остаток»; _ «4» – «До снятия»; _ «5» – «До даты»; _ «6» – «В течение сессии»; _ «7» – «Открытие»; _ «8» – «Закрытие»; _ «9» – «Кросс»; _ «11» – «До следующей сессии»; _ «13» – «До отключения»; _ «15» – «До времени»; _ «16» –«Следующий аукцион»
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
capacity	NUMBER	Роль в исполнении заявки. Возможные значения: _ «0» – не определено; _ «1» – Agent; _ «2» – Principal; _ «3» – Riskless principal; _ «4» – CFG give up; _ «5» – Cross as agent; _ «6» – Matched Principal; _ «7» – Proprietary; _ «8» – Individual; _ «9» – Agent for other member; _ «10» – Mixed; _ «11» – Market maker
passive_only_order	NUMBER	Поле для получения параметров по западным площадкам. Если имеет значение «0», значит значение не задано
visible	NUMBER	Видимое количество. Параметр айсберг-заявок, для обычных заявок выводится значение: «0».
awg_price	NUMBER	Средняя цена исполнения. Актуально, когда заявка выполнилась частями
expiry_time	NUMBER	Время окончания срока действия заявки в формате <ЧЧММСС>. Для GTT-заявок, используется вместе со сроком истечения заявки (Expiry)

Параметр	Тип	Описание
revision_number	NUMBER	Номер ревизии заявки. Используется, если заявка была заменена с сохранением номера
price_currency	STRING	Валюта заявки
ext_order_status	NUMBER	Расширенный статус заявки. Возможные значения: _ «0» (по умолчанию) – не определено; _ «1» – заявка активна; _ «2» – заявка частично исполнена; _ «3» – заявка исполнена; _ «4» – заявка отменена; _ «5» – заявка заменена; _ «6» – заявка в состоянии отмены; _ «7» – заявка отвергнута; _ «8» – приостановлено исполнение заявки; _ «9» – заявка в состоянии регистрации; _ «10» – заявка снята по времени действия; _ «11» – заявка в состоянии замены
accepted_uid	NUMBER	UID пользователя-менеджера, подтвердившего заявку при работе в режиме с подтверждениями
filled_value	NUMBER	Исполненный объем заявки в валюте цены для частично или полностью исполненных заявок
extref	STRING	Внешняя ссылка, используется для обратной связи с внешними системами
settle_currency	STRING	Валюта расчетов по заявке
on_behalf_of_uid	NUMBER	UID пользователя, от имени которого выставлена заявка
client_qualifier	NUMBER	Квалификатор клиента, от имени которого выставлена заявка. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «3» – Legal Entity
client_short_code	NUMBER	Краткий идентификатор клиента, от имени которого выставлена заявка
investment_decision_maker_qualifier	NUMBER	Квалификатор принявшего решение о выставлении заявки. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
investment_decision_maker_short_code	NUMBER	Краткий идентификатор принявшего решение о выставлении заявки

Параметр	Тип	Описание
executing_trader_qualifier	NUMBER	Квалификатор трейдера, исполнившего заявку. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
executing_trader_short_code	NUMBER	Краткий идентификатор трейдера, исполнившего заявку
settle_date	NUMBER	Дата расчетов. Для свопов дата расчетов первой части операции своп
settle_date2	NUMBER	Дата расчетов второй части операции своп
start_date	NUMBER	Дата, начиная с которой допускается совершение валютирования. Заполняется только для контрактов типа «FLEX FORWARD» (см. operation_type)
operation_type	NUMBER	Тип совершаемой операции. Возможные значения: _ «-1» – «NOT_DEFINED»; _ «0» – «SPOT»; _ «1» – «FORWARD»; _ «2» – «SWAP»; _ «6» – «NDF»; _ «7» – «FLEX FORWARD»
qty2	NUMBER	Количество второй части операции своп
value2	NUMBER	Объем второй части операции своп
visibility_factor	NUMBER	Видимая часть в общей сумме заявки в %
visible_repo_value	NUMBER	Сумма РЕПО видимой части (точность валюты расчетов заявки/инструмента)
trading_session	NUMBER	Идентификатор торговой сессии. Возможные значения: _ «0» – «Не определено»; _ «1» – «Основная сессия»; _ «2» – «Дополнительная сессия»; _ «3» – «Итоги дня»
price_entry_type	NUMBER	Тип ввода значения цены заявки. Возможные значения: _ «1» – «По цене»; _ «2» – «По доходности»; _ «3» - «По средневзвешенной цене»
lseccode	STRING	Код инструмента, являющийся приоритетным обеспечением
benchmark	STRING	Идентификатор индикативной ставки
external_qty	NUMBER	Внешнее количество

Параметр	Тип	Описание
upd_userid	STRING	Биржевой идентификатор пользователя, снявшего заявку
trading_session_date	NUMBER	Дата биржевой торговой сессии
z_spread	NUMBER	Z-спред по цене

4.6 Позиции участника по торговым счетам

Описание параметров таблицы «Позиции участника по торговым счетам»:

Параметр	Тип	Описание
firmid	STRING	Идентификатор фирмы
sec_code	STRING	Код инструмента
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: _ «0» – для обычных счетов, _ «1» – для счета обеспечения

4.7 Позиции участника по инструментам

Описание параметров таблицы «Позиции участника по инструментам»:

Параметр	Тип	Описание
firmid	STRING	Фирма
seccode	STRING	Код инструмента
openbal	NUMBER	Входящий остаток
currentpos	NUMBER	Текущий остаток
plannedposbuy	NUMBER	Количество инструментов в активных заявках на покупку
plannedpossell	NUMBER	Количество инструментов в активных заявках на продажу
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	Набор битовых флагов
brokerref	STRING	Комментарий, обычно: <код клиента>/<номер поручения>
firmid	STRING	Идентификатор дилер
account	STRING	Торговый счет
condition	NUMBER	Направленность стоп-цены. Возможные значения: _ «4» – «<=», _ «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. При превышении значения «9 223 372 036 854 775 807» поле принимает значение «-1», в этом случае рекомендуется использовать поле «trans_id_str»
trans_id_str	STRING	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: _ «1» – стоп-лимит; _ «2» – условие по другому инструменту; _ «3» – со связанной заявкой; _ «6» – тейк-профит; _ «7» – стоп-лимит по исполнению активной заявки; _ «8» – тейк-профит по исполнению активной заявки; _ «9» – тейк-профит и стоп-лимит
orderdate	NUMBER	Дата выставления
alltrade_num	NUMBER	Сделка условия
stopflags	NUMBER	Набор битовых флагов
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»

Параметр	Тип	Описание
sec_code	STRING	Код инструмента заявки
class_code	STRING	Код класса заявки
condition_sec_code	STRING	Код инструмента стоп-цены
condition_class_code	STRING	Код класса стоп-цены
canceled_uid	NUMBER	Идентификатор пользователя, снявшего стоп-заявку
order_date_time	TABLE	Время выставления стоп-заявки
withdraw_datetime	TABLE	Время снятия стоп-заявки
activation_date_time	TABLE	Дата и время активации стоп-заявки

4.9 Ограничения по клиентским счетам

Описание параметров фьючерсного лимита (таблица «Ограничения по клиентским счетам»):

Параметр	Тип	Описание
firmid	STRING	Идентификатор фирмы
trdaccid	STRING	Торговый счет
limit_type	NUMBER	Тип лимита. Возможные значения: _ «0» – «Денежные средства»; _ «1» – «Залоговые денежные средства»; _ «2» – «По совокупным средствам»; _ «3» – «Клиринговые денежные средства»; _ «4» – «Клиринговые залоговые денежные средства»; _ «5» – «Лимит открытых позиций на спот-рынке»; _ «6» – «Суммарные залоговые средства в иностранной валюте (в рублях)»; _ «7» – «Залоговые средства в иностранной валюте»
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

Параметр	Тип	Описание
accruedint	NUMBER	Накопленный 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
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
risk_level	NUMBER	Уровень риска клиента. Возможные значения: _ «0» (пусто, по умолчанию) – уровень риска не указан; _ «1» – КНУР (клиент с начальным уровнем риска); _ «2» – КСУР (клиент со стандартным уровнем риска); _ «3» – КПУР (клиент с повышенным уровнем риска); _ «4» – КОУР (клиент с особым уровнем риска)
go_without_orders	NUMBER	Гарантийное обеспечение без учета заявок, с учетом текущего реализованного риска (с учетом текущих цен) по риск-параметрам на начало сессии
go_planned	NUMBER	Гарантийное обеспечение без учета заявок с учетом текущих риск-параметров и рыночных данных, в ден. выражении. Отображается с точностью до 2 знаков после разделителя
indicative_varmargin	NUMBER	Индикативная вариационная маржа с учетом текущего индикативного курса валют, руб. (рассчитывается аналогично текущей индикативной вариационной марже, учитывает в т.ч. вариационную маржу по закрытым позициям). Рассчитывается торговой системой срочного рынка Московской Биржи. Отображается с точностью до 2 знаков после разделителя
net_option_value	NUMBER	Нетто стоимости опционов

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: _ «0» – не определен; _ «1» – основной счет; _ «2» – клиентские и дополнительные счета; _ «4» – все счета торг. членов
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
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

Параметр	Тип	Описание
session_status	NUMBER	Актуальный статус торговой сессии. Возможные значения: «0» – не определено; «6» – завершился основной клиринг
premium	NUMBER	Накопленная премия по опционам

4.11 Позиции по деньгам

Описание параметров таблицы «Позиции по деньгам»:

Параметр	Тип	Описание
currcode	STRING	Код валюты
tag	STRING	Код позиции
firmid	STRING	Идентификатор фирмы
client_code	STRING	Код клиента
openbal	NUMBER	Входящий остаток
openlimit	NUMBER	Входящий лимит
currentbal	NUMBER	Текущий остаток
currentlimit	NUMBER	Текущий лимит
locked	NUMBER	Заблокировано. Сумма средств, заблокированных под исполнение заявок клиента
locked_value_coef	NUMBER	Стоимость активов в заявках на покупку немаржинальных инструментов
locked_margin_value	NUMBER	Стоимость активов в заявках на покупку маржинальных инструментов
leverage	NUMBER	Плечо
limit_kind	NUMBER	Срок расчетов. Возможные значения: – положительные целые числа, начиная с «0», соответствующие срокам расчетов из таблицы «Позиции по деньгам»: «0» – Т0, «1» – Т1, «2» – Т2 и т.д.; – отрицательные целые числа – технологические лимиты (используются для внутренней работы системы QUIK)
wa_position_price	NUMBER	Средневзвешенная цена приобретения позиции
orders_collateral	NUMBER	Гарантийное обеспечение заявок

Параметр	Тип	Описание
positions_collateral	NUMBER	Гарантийное обеспечение позиций

4.12 Удаление позиции по денежным средствам

Описание параметров таблицы Удаление позиции по денежным средствам:

Параметр	Тип	Описание
currcode	STRING	Код валюты
tag	STRING	Код позиции
client_code	STRING	Код клиента
firmid	STRING	Идентификатор фирмы
limit_kind	NUMBER	Срок расчетов. Возможные значения: _ положительные целые числа, начиная с «0», соответствующие срокам расчетов из таблицы «Позиции по деньгам»: «0» – T0, «1» – T1, «2» – T2 и т.д.; _ отрицательные целые числа – технологические лимиты (используются для внутренней работы системы QUIK)

4.13 Удаление позиции по инструментам

Описание параметров таблицы Удаление позиции по инструментам:

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	Срок расчетов. Возможные значения: _ положительные целые числа, начиная с «0», соответствующие срокам расчетов из таблицы «Позиции по инструментам»: «0» – T0, «1» – T1, «2» – T2 и т.д.; _ отрицательные целые числа – технологические лимиты (используются для внутренней работы системы QUIK)

4.14 Удаление фьючерсного лимита

Описание параметров таблицы Удаление фьючерсного лимита:

Параметр	Тип	Описание
firmid	STRING	Идентификатор фирмы
limit_type	NUMBER	Тип лимита. Возможные значения: _ «0» – «Денежные средства»; _ «1» – «Залоговые денежные средства»; _ «2» – «По совокупным средствам»; _ «3» – «Клиринговые денежные средства»; _ «4» – «Клиринговые залоговые денежные средства»; _ «5» – «Лимит открытых позиций на спот-рынке»; _ «6» – «Суммарные залоговые средства в иностранной валюте (в рублях)»; _ «7» – «Залоговые средства в иностранной валюте»

4.15 Позиции по инструментам

Описание параметров таблицы «Позиции по инструментам»:

Параметр	Тип	Описание
sec_code	STRING	Код инструмента
trdaccid	STRING	Счет депо
firmid	STRING	Идентификатор фирмы
client_code	STRING	Код клиента
openbal	NUMBER	Входящий остаток
openlimit	NUMBER	Входящий лимит
currentbal	NUMBER	Текущий остаток
currentlimit	NUMBER	Текущий лимит
locked_sell	NUMBER	В продаже. Количество инструментов, заблокированное под исполнение заявок клиента на продажу
locked_buy	NUMBER	В покупке. Количество инструментов в активных заявках клиента на покупку

Параметр	Тип	Описание
locked_buy_value	NUMBER	Стоимость инструментов, заблокированных под покупку
locked_sell_value	NUMBER	Стоимость инструментов, заблокированных под продажу
wa_position_price	NUMBER	Цена приобретения
wa_price_currency	STRING	Валюта, в которой отображается значение цены приобретения инструмента. Для облигаций, цена которых выражена в процентах от номинала, возвращается значение «%». Параметр может принимать значение <пусто>, если в момент запроса данных в терминале отсутствует информация для расчета валюты цены приобретения
limit_kind	NUMBER	Срок расчетов. Возможные значения: _ положительные целые числа, начиная с «0», соответствующие срокам расчетов из таблицы «Позиции по инструментам»: «0» – T0, «1» – T1, «2» – T2 и т.д.; _ отрицательные целые числа – технологические лимиты (используются для внутренней работы системы QUIK)
locked_out	NUMBER	Количество ценных бумаг, заблокированных в поручениях на вывод

4.16 Позиции участника по деньгам

Описание параметров таблицы «Позиции участника по деньгам»:

Параметр	Тип	Описание
firmid	STRING	Идентификатор фирмы
currcode	STRING	Код валюты
tag	STRING	Код позиции
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

Параметр	Тип	Описание
ordersell	NUMBER	В заявках на покупку
netto	NUMBER	Нетто-позиция
plannedbal	NUMBER	Плановая позиция
debit	NUMBER	Дебет
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	Набор битовых флагов
brokerref	STRING	Комментарий, обычно: <код клиента>/<номер поручения>
userid	STRING	Трейдер
firmid	STRING	Идентификатор дилера
cpuserid	STRING	Трейдер 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

Параметр	Тип	Описание
value	NUMBER	Объем
price2	NUMBER	Цена выкупа
reporate	NUMBER	Ставка РЕПО (%)
refundrate	NUMBER	Ставка возмещения (%)
trans_id	NUMBER	Идентификатор транзакции. При превышении значения «9 223 372 036 854 775 807» поле принимает значение «-1», в этом случае рекомендуется использовать поле «trans_id_str»
trans_id_str	STRING	Идентификатор транзакции
client_code	STRING	Код клиента
repoentry	NUMBER	Тип ввода заявки РЕПО. Возможные значения: _ «0» – «Не определен»; _ «1» – «Цена1+Ставка»; _ «2» – «Ставка+ Цена2»; _ «3» – «Цена1+Цена2»; _ «4» – «Сумма РЕПО + Количество»; _ «5» – «Сумма РЕПО + Дисконт»; _ «6» – «Количество + Дисконт»; _ «7» – «Сумма РЕПО»; _ «8» – «Количество»
repovalue	NUMBER	Сумма РЕПО
repo2value	NUMBER	Объем выкупа РЕПО
repoterm	NUMBER	Срок РЕПО
start_discount	NUMBER	Начальный дисконт (%)
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_repovalue	NUMBER	Сумма РЕПО первоначальная
origin_qty	NUMBER	Количество первоначальное
origin_discount	NUMBER	Процент дисконта первоначальный

Параметр	Тип	Описание
neg_deal_activation_date	NUMBER	Дата активации заявки
neg_deal_activation_time	NUMBER	Время активации заявки
quoteno	NUMBER	Встречная безадресная заявка
settle_currency	STRING	Валюта расчетов
sec_code	STRING	Код инструмента
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
lseccode	STRING	Приоритетное обеспечение
canceled_uid	NUMBER	UID снявшего заявку
system_ref	STRING	Системная ссылка
price_currency	STRING	Валюта, в которой указана цена заявки
order_exchange_code	STRING	Биржевой номер заявки
extref	STRING	Внешняя ссылка, используется для обратной связи с внешними системами
period	NUMBER	Период торговой сессии, в которую была подана заявка
client_qualifier	NUMBER	Квалификатор клиента, от имени которого выставлена заявка. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «3» – Legal Entity
client_short_code	NUMBER	Краткий идентификатор клиента, от имени которого выставлена заявка

Параметр	Тип	Описание
investment_decision_maker_qualifier	NUMBER	Квалификатор принявшего решение о выставлении заявки. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
investment_decision_maker_short_code	NUMBER	Краткий идентификатор принявшего решение о выставлении заявки
executing_trader_qualifier	NUMBER	Квалификатор трейдера, исполнившего заявку. Возможные значения: _ «0» – не определено; _ «1» – Natural Person; _ «2» – Algorithm
executing_trader_short_code	NUMBER	Краткий идентификатор трейдера, исполнившего заявку
settle_date	NUMBER	Дата расчетов. Для свопов дата расчетов первой части операции своп
benchmark	STRING	Идентификатор индикативной ставки
ext_negdeal_flags	NUMBER	Набор битовых флагов. Возможные значения: _ бит 0 (0x1) – Валюты на адресной заявке заполнены в соответствии с общими правилами заполнения валют; _ бит 1 (0x2) – Признак заявки типа РЕПО с открытой датой
open_repo2date	NUMBER	Расчетная дата исполнения РЕПО с открытой датой
open_repo2value	NUMBER	Стоимость выкупа РЕПО с открытой датой
reject_reason	STRING	Причина отклонения
trading_session_date	NUMBER	Дата биржевой торговой сессии
filled_value	NUMBER	Исполненный объем заявки
valueentrytype	NUMBER	Тип ввода заявки
repovaluebalance	NUMBER	Остаток суммы РЕПО
repo2valuebalance	NUMBER	Остаток суммы выкупа РЕПО
open_repo_evergreen_term	NUMBER	Период до исполнения второй части сделки РЕПО с открытой датой
percent_period	NUMBER	Период процентных выплат

Параметр	Тип	Описание
z_spread	NUMBER	Z-спред по цене

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	Набор битовых флагов
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

Параметр	Тип	Описание
balance	NUMBER	Остаток
settle_time	NUMBER	Время исполнения
amount	NUMBER	Сумма обязательства
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: _ «0» – «Внесистемная сделка», _ «1» – «Первая часть сделки РЕПО», _ «2» – «Вторая часть сделки РЕПО», _ «3» – «Компенсационный взнос», _ «4» – «Дефолтер: отложенные обязательства и требования», _ «5» – «Пострадавший: отложенные обязательства и требования»
operation_type	NUMBER	Направленность. Возможные значения: _ «1» – «Зачислить», _ «2» – «Списать»
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

Параметр	Тип	Описание
settled	NUMBER	Состояние расчетов по сделке. Возможные значения: _ «1» – «Processed», _ «2» – «Not processed», _ «3» – «Is processing»
clearing_type	NUMBER	Тип клиринга. Возможные значения: _ «1» – «Not set», _ «2» – «Simple», _ «3» – «Multilateral»
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 инструмента
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 инструментов к исполнению (в лотах)
confirmtime	NUMBER	Время подтверждения в формате <ЧЧММСС>
ex_flags	NUMBER	Расширенные флаги сделки для исполнения. Возможные значения: _ «1» – «Подтверждена контрагентом»; _ «2» – «Подтверждена»
confirmreport	NUMBER	Номер поручения
extref	STRING	Внешняя ссылка, используется для обратной связи с внешними системами
benchmark	STRING	Идентификатор индикативной ставки
benchmark_change_date	NUMBER	Дата изменения индикатора в формате <ГГГГММДД>
benchmark_value	NUMBER	Значение индикатора, в %
cancel_reason	STRING	Причина отмены

Параметр	Тип	Описание
deposit_intent	NUMBER	Тип сделки депозита. Возможные значения: _ «0» – сделка не типа «депозит»; _ «1» – «Намерение»; _ «2» – «Депозит»; _ «3» – «Изъятие»; _ «4» – «Закрытие планового пополнения»; _ «5» – «Принудительное закрытие»; _ «6» – «Перенос пополнения»
open_repo2date	NUMBER	Расчетная дата исполнения РЕПО с открытой датой
open_repo2value	NUMBER	Стоимость выкупа РЕПО с открытой датой
open_repo_report_no	NUMBER	Отчет на закрытие сделки РЕПО с Открытой датой
open_repo_status	NUMBER	Статус РЕПО с открытой датой. Возможные значения: _ «0» – «Нет»; _ «1» – «Да»; _ «2» – «Срочная»
trading_session_date	NUMBER	Дата биржевой торговой сессии
open_repo_evergreen_term	NUMBER	Период до исполнения второй части сделки РЕПО с открытой датой
due_date	NUMBER	Надлежащая дата расчетов сделки
early_settle_status	NUMBER	Статус досрочного исполнения
early_settle_report_num	NUMBER	Отчет на досрочное исполнение
percent_period	NUMBER	Период процентных выплат
planned_interest	NUMBER	Сумма будущих процентных выплат
early_settle_cpreport_num	NUMBER	Отчет на досрочное исполнение партнера

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

Параметр	Тип	Описание
fullcoveredsell	NUMBER	Запрет необеспеченных продаж. Возможные значения: _ «0» – Нет; _ «1» – Да
main_trdaccid	STRING	Основной торговый счет
bankid_t0	STRING	Расчетная организация по «T0»
bankid_tplus	STRING	Расчетная организация по «T+»
trdacc_type	NUMBER	Тип торгового счета
depunitid	STRING	Раздел счета Депо
status	NUMBER	Статус торгового счета. Возможные значения: _ «0» – операции разрешены; _ «1» – операции запрещены
firmuse	NUMER	Тип раздела. Возможные значения: _ «0» – раздел обеспечения; _ иначе – для торговых разделов
depaccid	STRING	Номер счета депо в депозитарии
bank_acc_id	STRING	Код дополнительной позиции по денежным средствам
flags	NUMBER	Набор битовых флагов. Возможные значения: _ бит 0 (0x1) – Торгово-клиринговый счет для рынка депозитов

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	Набор битовых флагов
userid	STRING	Идентификатор пользователя
firmid	STRING	Идентификатор фирмы
account	STRING	Счет депо
cpfirmid	STRING	Код фирмы партнера
cpaccount	STRING	Код торгового счета партнера

Параметр	Тип	Описание
qty	NUMBER	Количество инструментов, в лотах
value	NUMBER	Объем сделки, выраженный в рублях
withdraw_time	NUMBER	Время снятия заявки
report_type	NUMBER	Тип отчета
report_kind	NUMBER	Вид отчета
commission	NUMBER	Объем комиссии по сделке, выраженный в руб
sec_code	STRING	Код инструмента
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
bsid	STRING	Bloomberg ID
cusip_code	STRING	CUSIP

Параметр	Тип	Описание
stock_code	STRING	StockCode
couponvalue	NUMBER	Размер купона
first_currcode	STRING	Код котируемой валюты в паре
second_currcode	STRING	Код базовой валюты в паре
base_active_classcode	STRING	Код класса базового актива
base_active_seccode	STRING	Базовый актив
option_strike	NUMBER	Страйк опциона
qty_multiplier	NUMBER	Кратность при вводе количества
step_price_currency	STRING	Валюта шага цены
sedol_code	STRING	SEDOL
cfi_code	STRING	CFI
ric_code	STRING	RIC
buybackdate	NUMBER	Дата оферты
buybackprice	NUMBER	Цена оферты
list_level	NUMBER	Уровень листинга
qty_scale	NUMBER	Точность количества
yieldatprevwaprice	NUMBER	Доходность по предыдущей оценке
regnumber	STRING	Регистрационный номер
trade_currency	STRING	Валюта торгов
second_curr_qty_scale	NUMBER	Точность количества котируемой валюты
first_curr_qty_scale	NUMBER	Точность количества базовой валюты
accruedint	NUMBER	Накопленный купонный доход
stock_name	STRING	Код деривативного контракта в формате QUIK
nextcoupon	NUMBER	Дата выплаты купона
couponperiod	NUMBER	Длительность купона
settlecode	STRING	Текущий код расчетов для инструмента
exp_date	NUMBER	Дата экспирации
settle_date	NUMBER	Дата расчетов
legs	TABLE	Ноги составного инструмента в формате leg_<N>

Параметр	Тип	Описание
_ leg_<N>	TABLE	Нога составного инструмента, где N может принимать целые значения от «0»
_ classcode	STRING	Параметр ноги, обозначающий код класса связанного инструмента
_ seccode	STRING	Параметр ноги, обозначающий код связанного инструмента
_ CFI	STRING	Параметр ноги, обозначающий CFI
_ ratio_qty	NUMBER	Параметр ноги, обозначающий количество образования позиции при выполнении операции по связанному инструменту
_ leg_side	STRING	Параметр ноги, обозначающий направление позиции при выполнении операции по связанному инструменту. Возможные значения: _ «buy» – покупка; _ «sell» – продажа
_ settle_date	NUMBER	Параметр ноги, обозначающий дату расчетов

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	Формат даты и времени
doesExist	NUMBER	Признак расчета индикатора при наличии свечки. Возможные значения: _ «0» – индикатор не рассчитан; _ «1» – индикатор рассчитан

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. При превышении значения «9 223 372 036 854 775 807» поле принимает значение «-1», в этом случае рекомендуется использовать поле «trans_id_str»
trans_id_str	STRING	Пользовательский идентификатор транзакции
status	NUMBER	Статус транзакции. Возможные значения: _ «0» – транзакция отправлена серверу; _ «1» – транзакция получена на сервер QUIK от клиента; _ «2» – ошибка при передаче транзакции в торговую систему. Так как отсутствует подключение шлюза Московской Биржи, повторно транзакция не отправляется;

Параметр	Тип	Описание
		_ «3» – транзакция выполнена; _ «4» – транзакция не выполнена торговой системой. Более подробное описание ошибки отражается в поле «Сообщение»; _ «5» – транзакция не прошла проверку сервера QUIK по каким-либо критериям. Например, проверку на наличие прав у пользователя на отправку транзакции данного типа; _ «6» – транзакция не прошла проверку лимитов сервера QUIK; _ «10» – транзакция не поддерживается торговой системой; _ «11» – транзакция не прошла проверку правильности электронной цифровой подписи; _ «12» – не удалось дождаться ответа на транзакцию, т.к. истек таймаут ожидания. Может возникнуть при подаче транзакций из QPILE; _ «13» – транзакция отвергнута, так как ее выполнение могло привести к кросс-сделке (т.е. сделке с тем же самым клиентским счетом); _ «14» – транзакция не прошла контроль дополнительных ограничений, установленных брокером; _ «15» – транзакция принята после нарушения дополнительных ограничений, установленных брокером; _ «16» – транзакция отменена пользователем в ходе проверки дополнительных ограничений, установленных брокером
result_msg	STRING	Сообщение
date_time	TABLE	Время отправки ответа на транзакцию сервером клиенту (заполняется сервером при отправке ответа на транзакцию клиентским приложениям)
uid	NUMBER	Идентификатор
flags	NUMBER	Флаги транзакции

Параметр	Тип	Описание
order_flags	NUMBER	Параметр заявки, обозначающий флаги. Возможные значения: «бит 2 (0x4)» (заявка на продажу, иначе – на покупку). Значение параметра применимо только для транзакций с параметром направления операции (купля / продажа), в остальных случаях параметр имеет значение «0»
server_trans_id	NUMBER	Идентификатор транзакции на сервере
*order_num	NUMBER	Номер заявки
*price	NUMBER	Цена
*quantity	NUMBER	Количество
*balance	NUMBER	Остаток
*firm_id	STRING	Идентификатор фирмы
*account	STRING	Торговый счет
*client_code	STRING	Код клиента
*brokerref	STRING	Комментарий
*class_code	STRING	Код класса
*sec_code	STRING	Код инструмента
*exchange_code	STRING	Биржевой номер заявки
error_code	NUMBER	Числовой код ошибки. Значение равно «0», если транзакция выполнена успешно
error_source	NUMBER	Источник сообщения. Возможные значения: _ «1» – Торговая система; _ «2» – Сервер QUIK; _ «3» – Библиотека расчета лимитов; _ «4» – Шлюз торговой системы
first_ordernum	NUMBER	Номер первой заявки, которая выставлялась при автоматической замене кода клиента. Используется, если на сервере QUIK настроена замена кода клиента для кросс-сделки
gate_reply_time	TABLE	Время отправки ответа на транзакцию шлюзом (заполняется шлюзом при отправке ответа на транзакцию серверу QUIK)
sent_local_time	TABLE	Дата и время отправки транзакции, локальное время клиента в UTC

Параметр	Тип	Описание
got_local_time	TABLE	Дата и время получения ответа на транзакцию, локальное время клиента в UTC
orders	TABLE	Заявки. Параметр добавляется в ответ на транзакцию только при наличии двух и более заявок, связанных с транзакцией
_ order_<N>	TABLE	Заявка, где N может принимать целые значения от «0»
_ first_ordernum	NUMBER	Параметр заявки, обозначающий номер первой заявки, которая выставлялась при автоматической замене кода клиента. Используется, если на сервере QUIK настроена замена кода клиента для кросс-сделки
_ order_num	NUMBER	Параметр заявки, обозначающий номер заявки
_ price	NUMBER	Параметр заявки, обозначающий цену
_ quantity	NUMBER	Параметр заявки, обозначающий количество
_ balance	NUMBER	Параметр заявки, обозначающий остаток
_ order_flags	NUMBER	Параметр заявки, обозначающий флаги. Возможные значения: «бит 2 (0x4)» (заявка на продажу, иначе – на покупку). Значение параметра применимо только для транзакций с параметром направления операции (купля / продажа), в остальных случаях параметр имеет значение «0»
_ firm_id	STRING	Параметр заявки, обозначающий идентификатор фирмы
_ account	STRING	Параметр заявки, обозначающий торговый счет
_ client_code	STRING	Параметр заявки, обозначающий код клиента
_ brokerref	STRING	Параметр заявки, обозначающий комментарий
_ exchange_code	STRING	Параметр заявки, обозначающий биржевой номер заявки
_ sec_code	STRING	Параметр заявки, обозначающий код инструмента
_ class_code	STRING	Параметр заявки, обозначающий код класса

* – параметр может иметь значение **nil**.

4.25 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 инструментов в сделках
qty_buy	NUMBER	Количество инструментов в заявках на покупку
qty_sell	NUMBER	Количество инструментов в заявках на продажу
netto	NUMBER	Нетто-позиция
debit	NUMBER	Дебет
credit	NUMBER	Кредит
sec_code	STRING	Код инструмента заявки
class_code	STRING	Код класса заявки
planned_covered	NUMBER	Плановая позиция T+
firm_use	NUMBER	Тип раздела. Возможные значения: _ «0» – торговый раздел; _ «1» – раздел обеспечения

4.26 Валюта: обязательства и требования по активам

Описание параметров Таблицы обязательств и требований по активам на валютном рынке:

Параметр	Тип	Описание
sec_code	STRING	Код инструмента
class_code	STRING	Код класса
firmId	STRING	Идентификатор фирмы
account	STRING	Торговый счет

Параметр	Тип	Описание
bank_acc_id	STRING	Идентификатор расчетного счета в НКЦ
date	NUMBER	Дата расчетов
debit	NUMBER	Размер денежных обязательств
credit	NUMBER	Размер денежных требований
value_buy	NUMBER	Сумма денежных средств в заявках на покупку
value_sell	NUMBER	Сумма денежных средств в заявках на продажу
margin_call	NUMBER	Сумма возврата компенсационного перевода
planned_covered	NUMBER	Плановая позиция T+
debit_balance	NUMBER	Размер денежных обязательств на начало дня, с точностью до 2 знаков после десятичного разделителя
credit_balance	NUMBER	Размер денежных требований на начало дня, с точностью до 2 знаков после десятичного разделителя

5. Описание битовых флагов

5.1 Флаги для таблиц «Заявки», «Заявки на внебиржевые сделки»

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
бит 3 (0x8)	Заявка лимитированная, иначе – рыночная
бит 4 (0x10)	Исполнить заявку 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)	Скрытая заявка
бит 8 (0x100)	Снять остаток
бит 9 (0x200)	Айсберг-заявка

Флаг установлен	Значение
бит 10 (0x400)	Заявка отклонена торговой системой
бит 20 (0x100000)	Поле «linkedorder» заполняется номером стоп-заявки

5.2 Флаги для таблиц «Сделки», «Сделки для исполнения»

Флаг установлен	Значение
бит 0 (0x1)	Маржинальная сделка
бит 2 (0x4)	Сделка на продажу, иначе – на покупку
бит 3 (0x8)	Сделка по айсберг-заявке
бит 4 (0x10)	Отмененная сделка («Состояние» — «О»), поле «canceled_datetime» заполняется датой и временем отмены сделки
бит 5 (0x20)	Пассивная сделка («Состояние» — «П»)
бит 6 (0x40)	Активная сделка («Состояние» — «А»)
бит 7 (0x80)	Первая часть операции своп
бит 8 (0x100)	Вторая часть операции своп

5.3 Флаги для таблицы «Отчеты по сделкам для исполнения»

Флаг установлен	Значение
бит 6 (0x40)	Отчет отправлен пользователем (статус «Отправлена»)
бит 7 (0x80)	Отчет получен пользователем от другого контрагента по сделке для исполнения (статус «Получена»)

При наличии обоих флагов статус будет «Отправлена и получена». Если не установлено ни одного флага, статус — «Статус неизвестен».

5.4 Флаги для таблицы «Обезличенные сделки»

Флаг установлен	Значение
бит 0 (0x1)	Сделка на продажу

Флаг установлен	Значение
бит 1 (0x2)	Сделка на покупку

Если флаги не установлены, направление сделки не определено.

5.5 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

5.6 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

Флаг установлен	Значение
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

5.7 Флаги для таблицы «Транзакции»

Флаг установлен	Значение
бит 20 (0x21)	Служебная транзакция
бит 21 (0x22)	Транзакция на выставление 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,…)

6.6 bit.test

Функция проверяет состояние указанного бита в значении. Возвращает **true**, если бит равен «1», и **false**, если бит равен «0».

Формат вызова:

BOOLEAN bit.test(NUMBER x, NUMBER n)

где:

- x – значение;
- n – номер бита. Нумерация битов начинается с «0».

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…**.

Версия Lua-машины, в которой выполняется запуск индикаторов, по умолчанию «5.3.5». Ее можно изменить в настройках Рабочего места QUIK в разделе «Lua скрипты» - «Версия Lua для скриптов-индикаторов», описание приведено в п. 2.10.7 Раздела 2 «Основные принципы работы».

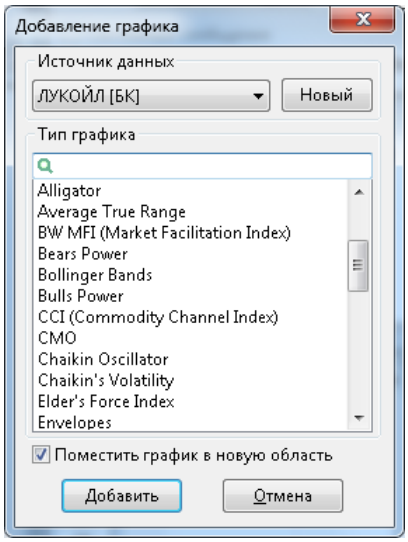
При добавлении нового индикатора на график плагин **qlua** сканирует папку **LuaIndicators**, проверяет файлы с расширением lua и luac (скомпилированные скрипты lua) на соответствие следующим требованиям:

- определена функция Init;
- определена функция OnCalculate;
- определена таблица Lua с именем Settings, в которой есть поле «Name»;
- luac файл скомпилирован под выбранную в настройках версию Lua-машины.

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_1 [, NUMBER v_n] OnCalculate(NUMBER index)

Параметры:

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

Если значение vi 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 OnChangeSettings

Функция вызывается при редактировании свойств индикатора после нажатия кнопок «Применить» или «ОК».

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

Формат вызова:

OnChangeSettings()

Пример:

```
Settings={Name="test1"}
function Init()
       return 1
end
function OnChangeSettings()
       message(Settings.Name)
end
```

7.2.5 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

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
```

7.2.6 CandleExist

Функция предназначена для проверки существования свечи на графике цены и объема:

BOOLEAN CandleExist(NUMBER index)

Параметры:

- **index** – индекс свечи.

Функция принимает в качестве параметра индекс свечи и возвращает «true» если свечка существует, иначе – «false».

> **Функция определяет свечи, которые не нужно учитывать в расчете индикатора. Такие свечи появляются в случае добавления на одну диаграмму с графиком цены и объема графика по истории значений параметра из Таблицы текущих торгов или при включенной настройке графика «Показывать пустые интервалы». Для таких свечей функции O, H, L, C, V возвращают «nil», а функция T – время пустой свечи.**

Пример использования функции для индикатора Moving Average по цене закрытия:

```
Settings= {
  Name = "Moving Average Lua",
  Period = 9,
  line =
  {
    {
      Name = "Moving Average Lua",
      Color = RGB(90, 110, 200),
      Type = TYPE_LINE,
```

```
      Width = 1
    }
  }
}

function Init()
  QUEUE = {}
  SUM = 0
  return 1
end
function OnCalculate(index)
  return Average(index)
end

function Average(indx)
  --если выполняется перерасчет индикатора, то сбрасываем очередь и сумму
  if indx == 1 then
    QUEUE = {}
    SUM = 0
  end
  --если свечка непустая, добавляем ее в очередь и суммируем значение ее закрытия
  if CandleExist(indx) then
    table.insert(QUEUE, {C_value = C(indx)})
    SUM = SUM + QUEUE[#QUEUE].C_value
    --если очередь достигла необходимого периода, то считаем среднее значение
    if #QUEUE == Settings.Period then
      local avg = SUM/Settings.Period
      SUM = SUM - QUEUE[1].C_value
      table.remove(QUEUE, 1)
      return avg
    end
  else
    return nil
  end
end
```

7.2.7 getDataSourceInfo

Функция предназначена для получения информации об источнике данных для индикатора.

TABLE info getDataSourceInfo()

Функция возвращает таблицу Lua с параметрами:

> **ВАЖНО! Для корректной работы функции getDataSourceInfo, 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 инструмента 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

7.2.8 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. Нумерация начинается с «1»;
- **line_number** – номер линии. Нумерация начинается с «1»;
- **value** – устанавливаемое значение. Параметр может иметь значение «nil».

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

Пример использования функции приведен ниже.

7.2.9 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
```

7.2.10 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
```

7.2.11 Таблица «Settings»

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

Список предопределенных 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 предопределенных констант: TYPE_LINE, TYPE_HISTOGRAM, TYPE_POINT, TYPE_DASHDOT, TYPE_DASH, TYPE_TRIANGLE_UP, TYPE_TRIANGLE_DOWN, TYPE_CANDLE, TYPE_BAR.

```
Settings.line[1].Type = TYPE_LINE --линии
Settings.line[2].Type = TYPE_HISTOGRAM --гистограммы
Settings.line[3].Type = TYPE_POINT --точки
Settings.line[4].Type = TYPE_DASHDOT --точка-тире
Settings.line[5].Type = TYPE_DASH --тире
Settings.line[6].Type = TYPE_TRIANGLE_UP --треугольник вверх
Settings.line[7].Type = TYPE_TRIANGLE_DOWN --треугольник вниз
Settings.line[8].Type = TYPE_CANDLE --свечи
Settings.line[9].Type = TYPE_BAR --бары
```

- 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.

> **ВАЖНО! Для обозначения того, что параметр индикатора является вещественным числом, необходимо дополнить его значение суффиксом вида <.N>. В противном случае параметр считается целочисленным и при редактировании свойств индикатора вручную доступен ввод только целого значения.**
> **Например:**

```
Settings.double_param1 = 1.0
Settings.double_param2 = 1.1
Settings.integer_param = 1
```

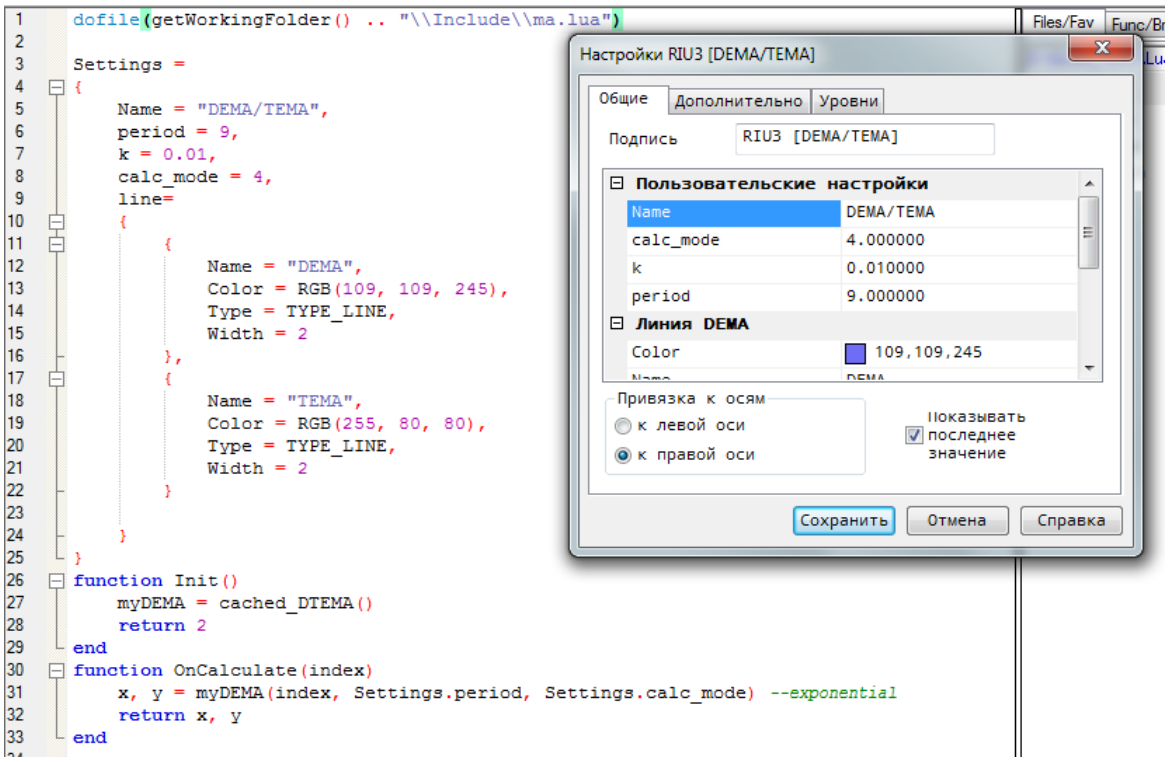
Поля, значения которых не определены в скрипте, будут проинициализированы значениями по умолчанию.

> **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».**

7.2.12 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.

7.2.13 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.

- **getParamEx2** – получает значения всех параметров биржевой информации из Таблицы текущих торгов с возможностью в дальнейшем отказаться от получения определенных параметров.
- **message** – отображает сообщения в терминале QUIK.
- **isDarkTheme** – позволяет получить информацию о том, в какой теме оформления интерфейса в настоящий момент работает терминал: стандартной или темной.
- **isConnected** – определяет состояние подключения клиентского места к серверу.
- **isUcpClient** – получает признак, указывающий имеет ли клиент единую денежную позицию.
- **PrintDbgStr** – функция для вывода отладочной информации.
- **getTradeDate** – получает дату торговой сессии.
- **getInfoParam** – позволяет получить параметры для информационного окна (Связь/Информационное окно).
- **getClassSecurities** – получает список кодов инструментов 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 инструменту.
- **getMoney** – получает информацию по денежным позициям.
- **getMoneyEx** – получает информацию по денежным позициям указанного типа.
- **getDepo** – получает позиции по инструментам.
- **getDepoEx** – получение позиций по инструментам указанного типа.
- **Subscribe_Level_II_Quotes** – заказывает на сервер получение стакана по указанному классу и инструменту.
- **Unsubscribe_Level_II_Quotes** – отменяет заказ на получение с сервера стакана по указанному классу и инструменту.
- **IsSubscribed_Level_II_Quotes** – функция позволяет узнать, заказан ли с сервера стакан по указанному классу и инструменту.
- **ParamRequest** – заказывает получение параметров Таблицы текущих торгов.
- **CancelParamRequest** – отменяет заказ на получение параметров Таблицы текущих торгов.
- **sendTransaction** – функция для работы с заявками.
- **CalcBuySell** – рассчитывает максимально возможное количество лотов в заявке.
- **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 срока расчетов.
- **getFuturesLimit** – получает информацию по фьючерсным лимитам.

- **getFuturesHolding** – получает информацию по фьючерсным позициям.
- **getBuySellInfoEx** – получает параметры таблицы «Купить/Продать» с учетом срока расчетов.
- **getTrdAccByClientCode** – возвращает торговый счет срочного рынка, соответствующий коду клиента фондового рынка с единой денежной позицией.
- **getTrdAccByClientCodeEx** – возвращает таблицу, содержащую все торговые счета, соответствующие коду клиента фондового рынка с единой денежной позицией.
- **getClientCodeByTrdAcc** – возвращает код клиента фондового рынка с единой денежной позицией, соответствующий торговому счету срочного рынка.
- **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.
- **getLinesCount** – получает информацию о количестве линий в графике (индикаторе) по выбранному идентификатору.
- **getNumCandles** – получает информацию о количестве свечек по выбранному идентификатору.
- **getCandlesByIndex** – получает информацию о свечках по идентификатору.

7.2.14Загрузка 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 строчку 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
```

```
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
	else
		-- возвращает строку, содержащую заголовок таблицы
```

```
                return GetWindowCaption(self.t_id)
        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)
                return true
        else
                SetCell(self.t_id, row, col_ind, tostring(data), data)
        end
```

```
end

function QTable:AddLine()
	-- добавляет в конец таблицы пустую строчку 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

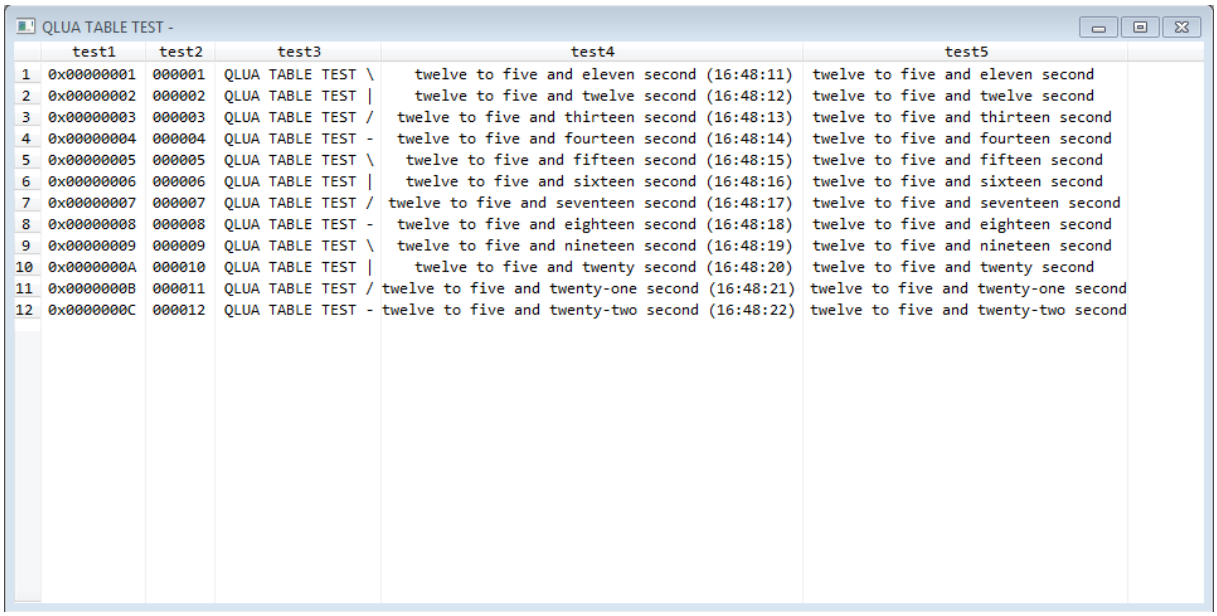
function _Time(t)
	hour = t.hour
	minute = t.min
	hour = hour % 12
	if(hour == 0) then
	hour, nextHourWord = 12, "one "
	else
	nextHourWord = numberToWord(hour+1)
	end
	hourWord = numberToWord(hour)
	if(minute == 0 ) then
	return hourWord .. "o'clock"
	elseif(minute == 30) then
```

```
        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
```

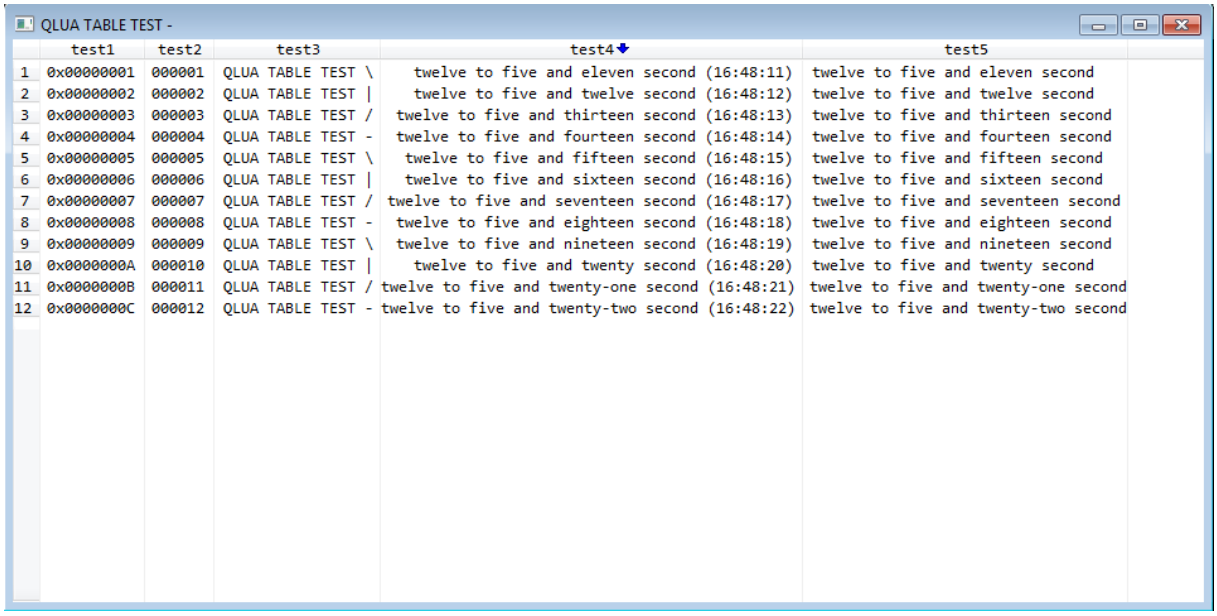
В результате выполнения скрипта в Рабочем месте QUIK создается таблица вида:



	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

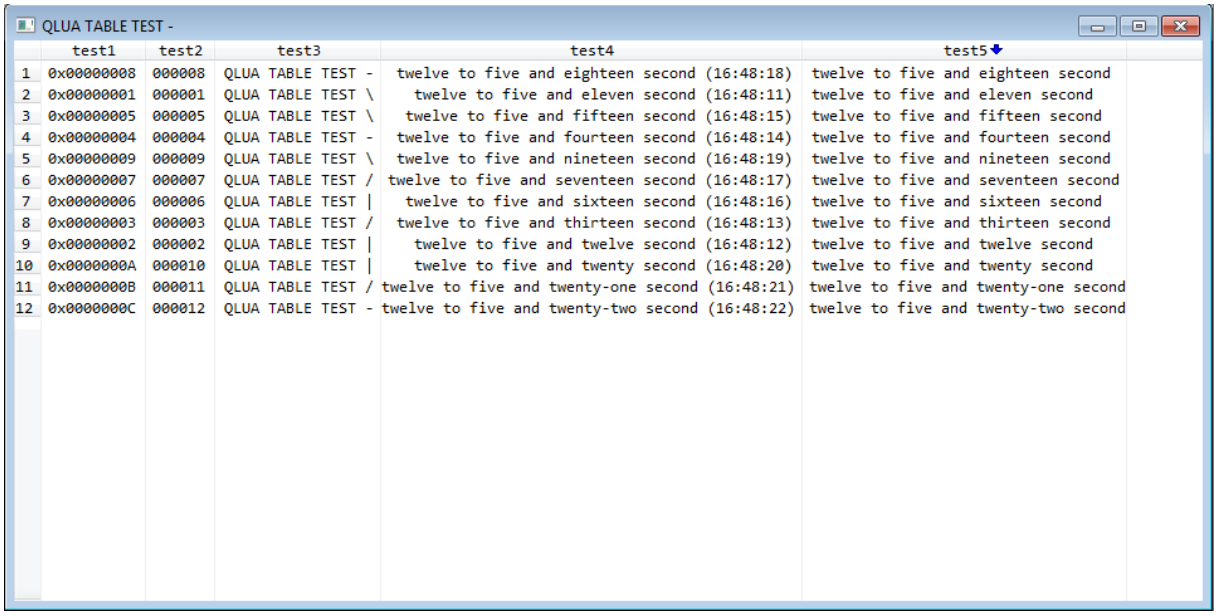
Приложение 2. Примеры сортировки в таблицах

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



QLUA TABLE TEST -

	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



QLUA TABLE TEST -

	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
```

```
            message("Event on click to cell 'DO NOT CLICK ME'!", 3)
      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, 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


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")

      while not stopped do
            sleep(100)
      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

---------------------
-- 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
```

```
-- 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

-- 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 = getPieceNoNil(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)
		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
       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

       -- 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
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)
        -- 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
```

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 примет 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
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")
```