ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ

Библиотеки доступа к торговой системе АО "Казахстанская фондовая биржа" из внешних приложений

Раздел 1. ОБЩИЕ ПОЛОЖЕНИЯ

В данном документе описываются принципы работы библиотеки доступа к торговой системе АО "Казахстанская фондовая биржа" (далее – Биржа) из внешних приложений (далее – Библиотека доступа).

Библиотека доступа реализована в виде динамической библиотеки client.dll. Для взаимодействия с указанной библиотекой не требуется программное обеспечение "Клиент" Торговой системы Биржи (далее – Терминал).

Библиотека доступа может устанавливаться на любом персональном компьютере, в том числе на том, где не установлен Терминал, однако в таком случае для ее корректной работы требуются дополнительные динамические библиотеки (dll), входящие в дистрибутив Терминала.

Обмен информацией с Торговой системой Биржи происходит путем вызова Библиотекой доступа функций внешних программ, т.е. без дополнительный действий со стороны пользователя.

Функции Библиотеки доступа позволяют внешним программам получать текущую информацию о параметрах финансовых инструментов и котировках по ним, сделках с финансовыми инструментами, заявках, репо–обязательствах, а также осуществлять подачу заявок в Торговую систему Биржи, подтверждений по сделкам и репо–обязательствам.

Внешние программы могут использовать следующие функции, описание которых приведено в разделе 2 "Функции" настоящего документа:

1) для инициализации работы Библиотеки доступа:

   global_init;

2) для получения информации по инструментам:

   CallMeFunction;

3) для получения информации о заявках:

   CallMeFunction2;

4) для получения информации о сделках:

   CallMeFunction3;

5) для получения информации об управляющих действиях, выяснения состояния передачи данных:

   CallMeFunction4;

6) для получения информации о котировках:

   CallMeFunction5;

7) для получения информации о репо–обязательствах:

   CallMeFunction6;

8) для получения информации по счетам:

   CallMeFunction7;

9) для передачи в торговую систему заявок, запросов на подачу и удаление заявок:

   put_order,

   del_order;

10) для передачи в торговую систему подтверждения по сделке:

    put_deal_confirm;

11) для передачи в торговую систему подтверждения по репо–обязательству:

put_repodeal_confirm;

12) для завершения сеанса работы с Библиотекой доступа:

session_terminate.

13) для получения информации о версии библиотеки:

get_curr_version.

14) для динамического осуществления подписки на инструменты:

add_market;

add_sector;

add_group;

add_instr;

delete_market;

delete_sector;

delete_group;

delete_instr.

Раздел 2. ФУНКЦИИ

В данном разделе приведено подробное описание функций, вызываемых внешними программными приложениями, условия их действия и результаты, возвращаемые ими.

Библиотека доступа позволяет получать информацию о следующих объектах Торговой системы:

1) инструменты;
2) заявки (все типы заявок);
3) сделки;
4) котировки;
5) репо–обязательства.

Информация, получаемая об объектах Торговой системы, содержит информацию об объектах, относящихся ко всем секторам рынков Торговой системы, доступных пользователю Терминала, под идентификатором и паролем которого было установлено соединение с Торговой системой.

При необходимости, внешняя программа через механизм подписки (см. далее аргументы функции global_init) может указать рынки, сектора, группы, или конкретные инструменты, по которым будет происходить обмен информацией.

2.1. Служебные функции

1) Инициализация Библиотеки доступа:

void __stdcall ***global_init***

(

char drHost,*	–	*адрес*
char drPort,*	–	*порт*
char drUser,*	–	*ник*
char drPassword,*	–	*пароль*
int isSSL,	–	*варианты подключения:* *0 – без SSL,* *1 – SSL с использованием ключевой дискеты,* *2 – SSL с использованием USB ключа*
CallMeFunction cmFunc,	–	*функция для получения информации по инструментам*
CallMeFunction2	–	*функция для получения информации по заявкам*

cmFunc2,

CallMeFunction3 cmFunc3, – *функция для получения информации по сделкам*

CallMeFunction4 cmFunc4, – *функция для получения отчетов по управляющим действиям*

CallMeFunction5 cmFunc5, – *функция для получения информации по котировкам*

CallMeFunction6 cmFunc6, – *функция для получения информации по репо–обязательствам*

CallMeFunction7 cmFunc7, – *функция для получения информации по счетам*

*OptionList *options =* – *структура для оформления подписки*

new OptionList

)

Данная функция производит подключение пользователя к удаленному серверу (по адресу и порту), регистрацию и авторизацию (по нику и паролю) с возможностью шифрования передаваемой информации (флаг isSSL), а также вызывает начальную загрузку данных по инструментам, котировкам, заявкам, репо–обязательствам и сделкам. Загрузка данных производится в указываемые пользователем функции, которые он указывает в качестве параметров, со следующими прототипами:

– void (__stdcall *CallMeFunction)(TExport* Inf) – прототип функции обратного вызова (CallBack) для получения инструментов;

– void (__stdcall *CallMeFunction2)(TExportOrder* Inf) – прототип функции обратного вызова (CallBack) для получения заявок;

– void (__stdcall *CallMeFunction3)(TExportDeal* Inf) – прототип функции обратного вызова (CallBack) для получения сделок;

– void (__stdcall *CallMeFunction4)(TExportErrorCode* Inf) – прототип функции обратного вызова (CallBack) для отчетов по управляющим действиям. В случае возникновения ошибки соединения с сервером, код ошибки содержится в поле SystemErrorCode;

– void (__stdcall *CallMeFunction5)(TInstrQuot* Inf) – прототип функции обратного вызова (CallBack) для получения котировок;

– void (__stdcall *CallMeFunction6)(TRepInfo* Inf) – прототип функции обратного вызова (CallBack) для получения репо–обязательств;

– void (_stdcall CallMeFunction7)(TExportTrdAcc* Inf) – прототип функции обратного вызова (CallBack) для получения счетов.

Пользователь также может указать в качестве параметра структуру Options, полями которой являются списки рынков, секторов, групп и инструментов, по которым будет осуществляться загрузка данных, а также интервал опроса сервера Библиотекой. Если данный аргумент пропущен, то загружаются все инструменты, а опрос происходит один раз в 5 секунд.

2) Завершение сеанса работы с Библиотекой доступа:

void __stdcall ***session_terminate*** *(void)*

Данная функция завершает сеанс работы с Библиотекой доступа и осуществляет выход пользователя из Торговой системы.

3) Получение отчетов по управляющим действиям:

*void (__stdcall **CallMeFunction4**)(TExportErrorCode* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 9–го параметра функции global_init, а затем вызывается Библиотекой доступа (в случае успешной подачи заявки на управляющее действие или ее отклонения, а также в случае возникновения ошибок соединения с сервером) с параметром Inf типа TExportErrorCode.

2.2. Инструменты

1) *Получение данных по инструментам:*

*void (__stdcall **CallMeFunction**)(TExport* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 6–го параметра функции global_init, а затем вызывается Библиотекой доступа (при наличии обновленной информации по инструментам) с параметром Inf типа TExport.

2) *Динамическая подписка на инструменты:*

*void __stdcall **add_market**(String market)*

Функция добавляет рынок *market* в список рынков в структуре *OptionList*.

*void __stdcall **delete_market**(String market)*

Функция удаляет рынок *market* из списка рынков в структуре *OptionList*.

*void __stdcall **add_sector**(String sector)*

Функция добавляет сектор *sector* в список секторов в структуре *OptionList*.

*void __stdcall **delete_sector**(String sector)*

Функция удаляет сектор *sector* из списка секторов в структуре *OptionList*.

*void __stdcall **add_group**(String group)*

Функция добавляет группу *group* в список групп в структуре *OptionList*.

*void __stdcall **delete_group**(String group)*

Функция удаляет группу *group* из списка групп в структуре *OptionList*.

*void __stdcall **add_instr**(String instr)*

Функция добавляет инструмент *instr* в список инструментов в структуре *OptionList*.

*void __stdcall **delete_instr**(String instr)*

Функция удаляет инструмент *instr* из списка интсрументов в структуре *OptionList*.

2.3. Заявки

1) *Получение данных по заявкам:*

*void (__stdcall **CallMeFunction2**)(TExportOrder* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 7–го параметра функции global_init, а затем вызывается Библиотекой доступа (при наличии обновленной информации по заявкам) с параметром Inf типа TExportOrder;

2) *Подача заявок:*

*void __stdcall **put_order**(Order Ord)*

Данная функция посылает запрос на подачу заявки в Торговую систему в соответствии со структурой Ord типа Order. При успешной подаче заявки Библиотекой доступа вызывается функция CallMeFunction4 (см. подпункт 3) пункта 2.1), в параметре которой поле Code имеет значение "0" (нуль). В случае отклонения заявки поле Code принимает ненулевое значение в соответствии с причиной отклонения (коды ошибок приведены в пункте 2.8 настоящего документа).

3) *Удаление заявок:*

*void __stdcall **del_order**(long InstrId, long OrderId, long ClntId)*

Данная функция посылает запрос на удаление заявки в Торговую систему по идентификатору инструмента и пользовательскому номеру заявки. Значение последнего параметра может быть произвольным. При успешном удалении заявки вызывается функция CallMeFunction4 (см. подпункт 3) пункта 2.1), в параметре которой поле Code имеет значение "0" (нуль). В случае отклонения удаления заявки поле Code принимает ненулевое значение в соответствии с причиной отклонения (коды ошибок приведены в пункте 2.8 настоящего документа).

2.4. Сделки

1) *Получение данных по сделкам:*

*void (__stdcall ***CallMeFunction3**)(TExportDeal* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 8–го параметра функции global_init, а затем вызывается Библиотекой доступа (при наличии обновленной информации по сделкам) с параметром Inf типа TExportDeal.

2) *Подача подтверждений по сделкам:*

*void __stdcall **put_deal_confirm**(TDealConfirm info)*

Данная функция посылает запрос на подачу подтверждения по сделке. Входящим параметром является структура, содержащая идентификатор инструмента, номер сделки, идентификатор действия и ID пользователя. Последний параметр допускается не указывать. При успешном выполнении действия по подтверждению вызывается функция CallMeFunction4 (см. подпункт 3) пункта 2.1), в параметре которой поле Code имеет значение "0" (нуль). В случае отклонения действия по подтверждению поле Code принимает ненулевое значение в соответствии с причиной отклонения (коды ошибок приведены в пункте 2.8 настоящего документа).

2.5. Котировки

Получение данных по котировкам:

*void (__stdcall ***CallMeFunction5**)(TInstrQuot* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 10–го параметра функции global_init, а затем вызывается Библиотекой доступа (при наличии обновленной информации по котировкам) с параметром Inf типа TInstrQuot.

2.6. Репо–обязательства

1) *Получение данных по репо–обязательствам:*

*void (__stdcall ***CallMeFunction6**)(TRepInfo* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 11–го параметра функции global_init, а затем вызывается Библиотекой доступа (при наличии обновленной информации по репо–обязательствам) с параметром Inf типа TRepInfo.

2) *Подача подтверждения по репо–обязательствам:*

*void __stdcall **put_repodeal_confirm**(TDealConfirm info)*

Данная функция посылает запрос на подачу подтверждения по сделке открытия репо–обязательства. Входящим параметром является структура, содержащая идентификатор инструмента, номер сделки открытия, идентификатор действия и ID пользователя. Последний параметр может не указываться. При успешном выполнении действия по подтверждению вызывается функция CallMeFunction4 (см. пункт 2.1 подпункт 3), в параметре которой поле Code имеет значение "0" (нуль). В случае отклонения действия по подтверждению поле Code принимает ненулевое значение в соответствии с причиной отклонения (коды ошибок приведены в пункте 2.8 настоящего документа).

2.7. Торговые счета

Получение информации по счетам:

*void (_stdcall **CallMeFunction7**)(TExportTrdAcc* Inf)*

Функция обратного вызова (callback) описывается в клиентском приложении пользователя и указывается в качестве 12–го параметра функции global_init, а затем вызывается Библиотекой доступа с параметром Inf типа TExportTrdAcc.

2.8. Коды состояний (статусы)

1) *Коды ошибок соединения с сервером представляют собой следующие возможные значения поля SystemErrorCode структуры TExportErrorCode:*

3000 – соединение разорвано. Попытка восстановить соединение;

3001 – установка соединения закончилась неуспешно;

3002 – отсутствует ключевая дискета;

3003 – вы можете работать только через SSL;

3004 – вы можете работать только через TCP;

3005 – срок действия старого пароля/ключей истек;

3006 – в данный момент такой пользователь уже работает в системе;

3007 – попытка использовать неактуальные ключи;

3008 – ошибка при создании SSL сессии;

3010 – Вашей организации заблокирован удаленный доступ.

2) ***Коды ошибок, возникающих при подаче/удалении заявок на управляющие действия (значения поля Code структуры TExportErrorCode):***

1000 – не хватает полномочий;

1400 – запрещена подача заявок;

1401 – запрещена подача заявок указанного типа;

1402 – неверное количество в заявке;

1403 – превышены лимиты;

1404 – нарушен минимальный шаг изменения цены;

1414 – инструмент заблокирован;

1501 – превышены лимиты по торговому счету;

1550 – превышены лимиты по денежному счету;

1551 – превышены лимиты по открытым позициям;

1700 – неправильный номер торгового счета в заявке;

1701 – номер чужого денежного счета;

1702 – нет прав на торговый счет;

1703 – нет прав на денежный счет;

2000 – неправильный ID инструмента.

Раздел 3. СТРУКТУРЫ ДАННЫХ, ИСПОЛЬЗУЕМЫХ ФУНКЦИЯМИ CLIENT.DLL

3.1. Структура заявки

1) ***Общая структура:***

```
struct TExportOrder
{
Order Order_data;
};
```

2) ***Структура данных заявки:***

```
typedef struct Order {
  long Id;                          – идентификатор инструмента
  char ShortName[ShortNameLen[1]];  – наименование инструмента
  long OrderId;                     – идентификатор заявки
  long OrderNum;                    – номер заявки, который может указать внешняя
                                      программа при ее подаче через функцию PutOrder)
```

[1] ShortNameLen = 25 – длина краткого наименования инструмента.

char Type;	–	*тип заявки:* *"1" – лимитированная,* *"2" – репо,* *"3" – прямая*
char Direction;	–	*направление:* *"1" – покупка,* *"2" – продажа,* *"0" – не определено*
double Price;	–	*цена*
long Quantity;	–	*количество*
double Volume;	–	*объем*
double RestVolume;	–	*остаток*
long Date;	–	*дата подачи*
long Time;	–	*время подачи*
long TrdAccId;	–	*торговый счет*
long ClntAcc;	–	*клиентский счет (счет, указываемый внешними программами)*
long Status;	–	*статус заявки согласно приложению 1 к настоящему документу*
char Firm[15];	–	*код контрагента*
char Currency;	–	*валюта:* *"0" – в тенге,* *"1" – в валюте торгов*
double ClosePrice;	–	*цена закрытия*
long CloseDate;	–	*дата закрытия*
long KredId;	–	*предмет репо – код инструмента (для заявок на рынке автоматического репо)*
long KredCnt;	–	*количество предмета репо (для заявок на рынке автоматического репо)*
char MM;	–	*маркет–мэйкеры*
int ClntId;	–	*ID пользователя (не указывается)*

};

3.2. Структура информации об инструменте

1) Общая структура:

struct TExport	–	*инструмент*
{		
int Cnt;	–	*количество пришедших бумаг*
char Type;	–	*тип пришедших данных:* *f – начальная загрузка,* *u – обновленный инструмент,* *a – добавленный инструмент,* *r – удаленный инструмент*
*TExport_data *Export_data[2000];*		*структура, содержащая данные по каждому инструменту*

};

2) *Структура данных об инструменте:*

struct TExport_data

{

int ID;	–	*идентификатор инструмента*
char MarketName;*	–	*наименование рынка*
char SectorName;*	–	*наименование сектора*
char GroupName;*	–	*наименование группы*
char ShortName;*	–	*краткое наименование инструмента*
char Name;*	–	*полное наименование инструмента*
char Status;	–	*состояние торгов по инструменту согласно приложению 2 к настоящему документу*
char SesNum;	–	*номер сессии*
float Open;	–	*курс открытия*
float Ask;	–	*цена предложения*
float Bid;	–	*цена покупки*
float Last;	–	*цена последней сделки*
int LastVolume;	–	*объем последней сделки в инструменте*
float Average;	–	*средняя цена*
float Max;	–	*максимальный курс*
float Min;	–	*минимальный курс*
int Volume;	–	*объем торгов в инструменте*
float VolTotal;	–	*объем торгов в контрвалюте*
int OrderCnt;	–	*количество заявок*
int DealCnt;	–	*количество сделок*
float UstKurs;	–	*официальный курс*
float KaseKurs;	–	*курс KASE*
int CurrId;	–	*ID валюты торгов*
char NIN;*	–	*НИН инструмента*
char IsCupon;	–	*вид бумаги:* *"0" – дисконтная,* *"1" – купонная*
int Nominal;	–	*номинал*
char Method;	–	*база исчисления ценной бумаги*
char Base*	–	*база дат выплат купона*
float CuponTax;	–	*купонная ставка*
int CuponCnt;	–	*количество выплат в году*
int PastPayDate;	–	*дата последней выплаты купон*
int NextPayDate;	–	*дата следующей выплаты купона*
int XDate;	–	*дата, с которой не начисляется накопленный интерес*

float AccInt;	–	*текущий накопленный интерес*
float StartKurs;	–	*курс доллара США на дату начала обращения*
float Koeff;	–	*коэффициент индексации курса доллара США*
int Days;	–	*количество дней до погашения*
int CloseDate;	–	*количество дата закрытия*
int OpenDate;	–	*дата открытия*
int Lot;	–	*лот*
float KorrCnt;	–	*множитель количества*
float KorrPrc;	–	*множитель цены*
int VisPrec;	–	*количество знаков после запятой*
bool IsBlocked;	–	*признак того, заблокирован ли инструмент*
char KursMethod;		

};

3.3. Структура информации о сделке

struct TExportDeal

{

long DealId;	–	*порядковый номер сделки*
long Id;	–	*идентификатор инструмента*
char ShortName[ShortNameLen];	–	*наименование инструмента*
char BS;	–	*направление сделки:* *"1" – покупка,* *"2" – продажа,* *"0" – не определено*
double Price;	–	*цена сделки*
double Volume;	–	*объем сделки*
long Date;	–	*дата подачи*
long TIme;	–	*время подачи*
long TrdAccId;	–	*торговый счет*
long ClntAcc;	–	*клиентский счет*
long OrderId;	–	*код заявки*
char NIN[15];	–	*НИН инструмента*
double AccInt;	–	*накопленный интерес*
long Days;	–	*дней до погашения*
char Firm[15];	–	*код контрагента*
double Yield;	–	*доходность*
long OpenDate;	–	*дата открытия*
long CloseDate;	–	*дата закрытия*
long Quantity;	–	*количество*
double ClosePrice;	–	*цена закрытия*
long KredId;	–	*предмет репо – код инструмента*
long KredCnt;	–	*количество предмета репо*

long OrderNum;	–	*номер заявки*
char Status;	–	*статус сделки согласно приложению 3 к настоящему документу*
long SystemId;	–	*системный ID*
char UserNick[15];	–	*ник пользователя*
char Type;	–	*тип сделки:* *"1" – простая,* *"2" – репо–открытия,* *"3" – прямая,* *"4" – репо–закрытия*
char MM;	–	*маркет–мэйкеры*
long SettlDate;	–	*дата расч.*
char KredNIN[15];	–	*НИН предмета репо*

};

3.4. Структура подписки на инструменты

struct OptionList	–	*выбор групп и инструментов, по которым осуществляется получение информации*
{		
*vector<String> * Markets;*	–	*список рынков, по которым будет приходить обновление*
*vector<String> * Sectors;*	–	*список секторов*
*vector<String> * Groups;*	–	*список групп*
*vector<String> * Instruments;*	–	*список инструментов*
int Time;	–	*периодичность опроса сервера (в секундах)*
};		

3.5. Структура отчета об управляющих действиях

struct TExportErrorCode	–	*сообщение об ошибке, непустое только при Type = e*
{		
long SystemErrorCode;	–	*код ошибки при соединении с сервером*
long OrderId;	–	*пользовательский ID заявки*
long OrderStatus;	–	*статус заявки:* *"0" – подача,* *"1" – удаление*
long ClntAcc;	–	*ID пользователя*
long Code;	–	*код ошибки при подаче/удалении заявки*
};		

3.6. Структура репо–обязательства

struct TRepInfo		
{		
long CloseDate;	–	*дата закрытия*
long Id;	–	*идентификатор инструмента*

double ClosePrice; – *цена закрытия*

long Quantity; – *количество*

char SellTrdAcc[15] – *торговый счет продавца*

char BuyTrdAcc[15]; – *торговый счет покупателя*

double OpenPrice; – *цена открытия*

long OpenDate; – *дата открытия*

long OpenDealId; – *код сделки открытия*

long CloseDealId; – *код сделки закрытия*

char SellConfirm; – *готовность продавца к исполнению сделки закрытия*

char BuyConfirm; – *готовность покупателя к исполнению сделки закрытия*

long KredId; – *предмет репо – код инструмента*

long KredCnt; – *количество предмета репо*

double KredOpenPrice; – *цена открытия по предмету репо*

double KredClosePrice; – *цена закрытия по предмету репо*

double CloseVol; – *объем закрытия*

};

3.7.Структура котировки

struct TInstrQuot – *котировка по данному инструменту*

{

char Type; – *тип записи:*
u – обновленная котировка;
a – добавленная котировка;
d – удаленная котировка

int InstrID; – *количество записей*

double Price; – *цена*

long BuyVolume; – *объем покупки*

long SellVolume; – *объем продажи*

long RemVolume; – *остаточный объем*

};

3.8.Структура подтверждения по сделке

struct TDealConfirm – *подтверждение сделки*

{

long Id; – *ID инструмента*

long DealId; – *ID сделки*

char ToDo; – *действие:*
"1" – подтвердить,
– *"2" – отклонить*

long ClntId; – *ID пользователя (не указывается)*

};

3.9.Структура счетов

```
struct TExportTrdAcc
{
  long TrdAccId;          – наименование счета
  char TrdAcc[15];        – идентификатор счета
};
```

Приложение 1
к Принципам функционирования
Библиотеки доступа к Торговой системе
АО "Казахстанская фондовая биржа"
из внешних приложений

Статусы заявок

1 – L – osLimit

2 – T – osTrade

3 – LT – osLimit+ osTrade

8 – u – osUserRemoved

9 – Lu – osUserRemoved + osLimit

10 – Tu – osUserRemoved + osTrade

11 – LTu – osUserRemoved + osLimit + osTrade

16 – o – osOperRemoved

17 – Lo – osOperRemoved + osLimit

18 – To – osOperRemoved + osTrade

19 – Lto – osOperRemoved + osLimit+ osTrade

32 – s – osSysRemoved

33 – Ls – osSysRemoved + osLimit

35 – LTs – osSysRemoved + osLimit+ osTrade

64 – D – osDeal

65 – LD – osDeal + osLimit

66 – TD – osDeal + osTrade

67 – LTD – osDeal + osLimit + osTrade

72 – uD – osDeal + osUserRemoved

73 – LuD – osDeal + osUserRemoved + osLimit

74 – TuD – osDeal + osUserRemoved + osTrade

75 – LTuD – osDeal + osUserRemoved + osLimit + osTrade

80 – oD – osDeal + osOperRemoved

81 – LoD – osDeal + osOperRemoved + osLimit

82 – ToD – osDeal + osOperRemoved + osTrade

83 – LToD – osDeal + osOperRemoved + osLimit + osTrade

96 – sD – osDeal + osSysRemoved

97 – LsD – osDeal + osSysRemoved + osLimit

99 – LTsD – osDeal + osSysRemoved + osLimit + osTrade

136 – ub – osBrokRemoved + osUserRemoved

137 – Lub – osBrokRemoved + osUserRemoved + osLimit

138 – Tub – osBrokRemoved + osUserRemoved + osTrade

139 – LTub – osBrokRemoved + osUserRemoved + osLimit + osTrade

200 – uDb – osBrokRemoved + osDeal + osUserRemoved

201 – LuDb – osBrokRemoved + osDeal + osUserRemoved + osLimit

202 – TuDb – osBrokRemoved + osDeal + osUserRemoved + osTrade

203 – LTuDb – osBrokRemoved + osDeal + osUserRemoved + osLimit + osTrade

265 – LuW – osWaitsConf + osUserRemoved + osLimit

266 – TuW – osWaitsConf + osUserRemoved + osTrade

273 – LoW – osWaitsConf + osOperatorRemoved + osLimit

274 – ToW – osWaitsConf + osOperatorRemoved + osTrade

393 – LubW – osWaitsConf + osBrokRemoved + osUserRemoved + osLimit

394 – TubW – osWaitsConf + osBrokRemoved + osUserRemoved + osTrade

529 – LoR – osRejectedConf + osOperatorRemoved + osLimit

530 – ToR – osRejectedConf + osOperatorRemoved + osTrade,

где:

osLimit – лимитированная заявка

osTrade – рыночная заявка

osUserRemoved – заявка удалена пользователем

osOperRemoved – заявка удалена оператором

osSysRemoved – заявка удалена системой

osDeal – по заявке заключена сделка

osBrokRemoved – заявка удалена брокером

osWaitsConf – заявка ожидает подтверждения

osRejectedConf – заявка отвергнута подтверждателем

Приложение 2
к Принципам функционирования
Библиотеки доступа к Торговой системе
АО "Казахстанская фондовая биржа"
из внешних приложений

Статусы торгов

1 – C – Закрыт

3 – T – Непрерывный встречный аукцион

5 – t – Непрерывный встречный аукцион (приостановлен)

11 – x – предварительный фиксинг

12 – X – фиксинг

18 – F – Франкфуртский аукцион (открытое размещение)

19 – U – Аукцион по цене отсечения

20 – A – Аукцион по заявленной цене

21 – f – Франкфуртский аукцион (приостановлен)

22 – u – Аукцион по цене отсечения (приостановлен)

23 – a – Аукцион по заявленной цене (приостановлен)

24 – P – Предварительные торги

25 – F – Франкфуртский аукцион

26 – F – Франкфуртский аукцион

Статусы сделок

0 – dsConfirmed – "Done"

1 – dsRejectedConf – "Rejected by Investor"

2 – dsRejectedPart – "Rejected by Partner"

3 – dsRejectedSys – "Rejected by CD"

4 – dsWaitConf – "Waits for Confirm"

5 – dsWaitPart – "Waits for Partner"

6 – dsWaitSys – "Waits for CD"

7 – dsWaitsBuyer – "Waits for Buyer"

8 – dsPaidBuyer – "Paid by Buyer"

9 – dsUnpaid – "Unpaid"

10 – dsNoteDelivered – "Note Delivered"

11 – dsUndelivered – "Undelivered"

12 – dsWaitsChange – "Waits for change note"

13 – dsWaitsAgree – "Waits for agreement"

14 – dsRejectDepo – //на клиенте не встречается

15 – dsRejectedSysNoMoney – "Rejected by CD(cash problem)"

16 – dsRejectedSysNoSecur – "Rejected by CD(sec.problem)"

17 – dsWaitsAgreeDoubleNoMoney – "Waits for double agreement(cash problem)"

18 – dsWaitsAgreeDoubleNoSecur – "Waits for double agreement(sec.problem)"

19 – dsNoAgreementNoMoney – "No agreement(cash problem)"

20 – dsNoAgreementNoSecur – "No agreement(sec.problem)"

21 – dsRejectRepo – "No agreement(chg repo)"