

---

## **AUXILIARY DOCUMENTATION FOR THE PROGRAMMING OF K2**

**Title:** Auxiliary documentation for programming  
**Prepared by:** IT Development Department  
**Reviewed by:**  
**Version:** 1.2  
**Date:** 16 May 2002

**File name:**Programozasi segédlet.doc

# Contents

<b>1.</b>	<b><i>Introduction</i></b>	<b>3</b>
<b>2.</b>	<b><i>Related documents</i></b>	<b>3</b>
<b>3.</b>	<b><i>Useful information</i></b>	<b>3</b>
<b>3.1.</b>	<b>What can a K2 client do? What is the role of the client?</b>	<b>4</b>
<b>4.</b>	<b><i>Advice in respect of programming / K2 client development</i></b>	<b>4</b>
<b>4.1.</b>	<b>Supported server and client platforms</b>	<b>4</b>
<b>4.2.</b>	<b>Entry of orders</b>	<b>5</b>
4.2.1.	Status of orders in K2	5
4.2.2.	Status of orders in the MMTS system	6
4.2.3.	K2 allocates a serial number to the orders entered	6
4.2.4.	Specific marking/identification of entered orders	6
4.2.5.	Checking the structure and content of order entries	7
4.2.6.	Trading account numbers necessary for order entries	7
4.2.7.	Handling of the price and yield fields upon entering orders	8
4.2.8.	Handling of the securities board (BoardID), security (SecID) and SecBoardID identifiers	8
<b>4.3.</b>	<b>Information queries from K2, types of information</b>	<b>9</b>
4.3.1.	Types of K2 client data queries	9
4.3.2.	Quicker client queries	9
4.3.3.	Return values of the data query functions	10
4.3.4.	Change numbers by boards	10
<b>4.4.</b>	<b>Client review: what kind of a K2 server is it connected to?</b>	<b>10</b>
<b>4.5.</b>	<b>Processing of a K2 response</b>	<b>11</b>
<b>4.6.</b>	<b>Review of error values and error messages after each client function call</b>	<b>11</b>
<b>4.7.</b>	<b>K2 server: daily booting, interruption of connection, normal shutdown at the end of the day, shutdown in case of errors</b>	<b>11</b>
4.7.1.	The normal daily booting of K2	11
4.7.2.	Restarting K2 during the day	12
4.7.3.	Normal shut-down of K2 at the end of the day	12
<b>4.8.</b>	<b>Multiple K2 client processes and thread programming</b>	<b>13</b>
<b>4.9.</b>	<b>Recommended daily cycle for communication with K2</b>	<b>13</b>
<b>4.10.</b>	<b>Tracing on the client's side</b>	<b>13</b>
<b>4.11.</b>	<b>Sample programs, useful utilities</b>	<b>13</b>
<b>4.12.</b>	<b>Paying attention to new securities and changes to other trading parameters</b>	<b>14</b>
<b>5.</b>	<b><i>Other information on the daily operation of K2</i></b>	<b>14</b>
<b>5.1.</b>	<b>Booting K2 in the morning and its shut-down in the evening</b>	<b>14</b>
<b>5.2.</b>	<b>Manual setting of K2 client user data</b>	<b>14</b>
<b>5.3.</b>	<b>Periodical setting of the time with K2</b>	<b>14</b>

# Index of modifications

Version	Author	Date	Summary of modifications
1.0	Ferenc Pittner	9 May 2002	First version
1.1	Attila Vass	16 May 2002	Supplements
1.2	Ferenc Pittner	29 September 2004	Corrections

## 1. Introduction

The Budapest Stock Exchange (hereinafter: the BSE) purchased the source code for K2 connection server software along with the rights to the modification and sales thereof, from Effice Mérnöki Iroda Kft. in May 2002. As a result of the transaction, it became possible for the BSE to sell and maintain K2 software. This document is an attempt to provide programming assistance for all brokerage firms planning to develop a client connection interface for K2 software.

This document was prepared because neither the original description of the interface prepared by the authors nor the K2 User Manual drew developers' attention to certain details which we deem indispensable for the precise planning of the client software's development.

BSE personnel may later modify the documentation prepared by Effice and the information contained herein may be incorporated into the User Manual.

We recommend that K2 client developers read both this document and the K2 User Manual thoroughly before engaging in any development work. It is important that before designing K2 client software they understand the logic of the operation of K2, since this is the only way to ensure the development of software which operates securely under all circumstances.

Herein we have summarised useful pieces of information and set out our recommendations in respect of the operation of the K2 client - recommendations which, of course, are not of a mandatory nature. This document was prepared by BSE personnel based on their experience gained during the testing and trial of K2.

## 2. Related documents

The following related documentation exists at the time of the preparation of this document:

- K2/ifsc User Manual (v2.2/2001) - Effice
- MMTSI data types (v1.1/2001) – Effice
- MMTSII data types (v1.1/2001) – Effice

## 3. Useful information

At the time of the preparation of this document, K2 is the only physical implementation of the Connection Server defined in the Regulations on Remote Trading. It enables

automated/programmable communication between the broker's IT (front office/back office) system and the BSE MMTS central trading system.

### **3.1. *What can a K2 client do? What is the role of the client?***

The **K2 server** runs on a separate computer and creates on one hand connection with the MMTS and on the other hand the K2 client application developed/purchased by the brokerage firm. K2 queries the majority of trading data from the central server and forwards the orders arriving from the K2 client to the central trading system. Although several K2 clients may be connected to the K2 server from the broker's system (see below), the K2 server logs into the BSE central system as a single user and the orders entered by K2 clients arrive at the central trading system in the name of this single user.

The **K2 client** is an application/software which the broker either develops himself or has developed by someone. The client software is connected to the K2 server and maintains the connection with it. The K2 client may be an integral part of the broker's own system, which is prepared for maintaining the connection with K2. The documentation listed in Section 2 defines the K2 client as an "IFSC" (Interface Socket Client), while in several cases it is referred to as a User (U). The K2 client is able to maintain a connection with the K2 server through the routine libraries (DDL or LIB) translated into various platforms.

Due to the characteristics of the BSE central trading system, there are two physically separate K2 servers (a cash and a derivative module), which actually means two programs running parallel to each other. However, on the client-side there is only one routine library and the client software only defines whether it wants to connect to the cash or the derivative K2 server program upon connecting. Indeed, it can connect to both systems at the same time.

Basically, the K2 client can send orders to the central trading system through the K2 server, and it can query market data from the K2 server. The connection typically has a "query-respond" nature since the K2 server only responds to the queries of the client and it does not initiate the sending of any data automatically. Multiple client applications may be connected to a K2. However, there are some restrictions in respect of this matter (see 4.8, Multiple K2 client processes and thread programming).

## **4. Advice in respect of programming / K2 client development**

### **4.1. *Supported server and client platforms***

At present, the K2 server is accessible under HP UNIX and LINUX. Of course, the Linux environment is less expensive. Consequently, the BSE is prepared for sales of this version. The targeted Linux type is RedHat.

The developer may select from several client-side solutions:

- Windows NT;

- Linux (RedHat)
- HP-UX

On the client-side, a function library is available for the developers (along with the necessary related files). While under Window, this means a DLL and some header files, under UNIX/Linux it implies a LIB library and header files.

The function library may be translated into other platforms as well, but this can only take place based on a specific request and the result cannot be guaranteed.

## **4.2. Entry of orders**

It is important to review in detail and understand the process of entering orders. Despite the fact that the K2/ifsc User Manual refers to this topic in some detail, we believe it is important to consider the following:

### **4.2.1. Status of orders in K2**

- a. The entry, modification, and cancellation of orders is, of course, initiated by the K2 client.
- b. These requests are first transferred from the client to K2, and the client receives confirmation and an ID in respect of this. (Possible status of the order in K2: *accepted* – it is waiting for approval; *confirmed* – no approval is necessary; *denied*-erroneous order content).
- c. If the order must be approved by another client, then it will be waiting in K2 until this approval is granted (K2 will not send it to the central trading system until then). (Possible status of the order in K2 after the transaction: *confirmed* – approval given; *denied*-not approved).
- d. The K2 server checks in cycles whether there is anything to do in respect of this, and if it finds an order on hold, then it will send it to the central trading system and is likely (note!) to receive an answer for this. (Possible status of the order in K2: *unknown* – order sent to the trading system but no answer has been received yet; *refused* – the central system rejected the order for some reason; *entered* – successful entry of the order into the central trading system).
- e. The client will not receive feedback on this automatically (see paragraph d), but it must query order table itself. If it finds new data, it must process those and must monitor the changes to the status of orders on a continuing basis (this task is not too complicated, since the registration of status changes takes place by handling the change numbers, see Section 4.3.4 ).

In reality, another situation may arise which is in a way between the events described in paragraphs d and e: if K2 stops right after sending the order to the central system for any reason i.e. some orders remain in “U” status (this means that K2 tried to send the order to the central system before the failure, but the action was either unsuccessful or K2 did not receive the answer). In this case, K2 will not do anything to these orders upon restart, therefore, the client must decide whether to put them into the Confirmed status (in this case,

K2 will send them again to the central trading system) or the Denied status (in this case, K2 will not send them to the central trading system) – see Section 4.7.2: Restarting K2 during the day.

Here we note that K2 registers the changes to the status of orders in separate files, which can be viewed with the rldump1 and rldump2 utility programs (see Section 4.11: Sample programs). This is extremely useful upon testing.

It is also important to note that the status of orders is also set by K2; this can also be set by the client application itself (c).

The definitions used in the User Manual are rather confusing. For example, it is very important to know that the client may query orders from K2 from two places/tables:

- K2's own order table (e.g. Table 11 Order Entry Record, IFSC User Manual), containing orders sent by the K2 client to the K2 server;
- the orders received from the MMTS central system (the other K2order board, e.g. Table 8 Order Record, IFSC User Manual).

The first board contains the aforementioned status and the K2 identifier mentioned in paragraph b (e.g. the STATUS and ORDERID fields of Table 11).

#### 4.2.2. Status of orders in the MMTS system

Naturally, the central trading system allocates a specific status to the orders (e.g. OPEN, AMENDED, MATCHED etc. see MMTS I. and MMTS II. data types – Section 2: Related documents). For example, the status given by the central trading system can be found in the STATUS field of Table 8.

#### 4.2.3. K2 allocates a serial number to the orders entered by the K2 client

K2 will allocate a serial number to the order entered in paragraph a – if it was successfully transferred to K2 (new\_orderid field of the ifsc\_orderentry function). This is important for two reasons:

- if the status of the order changes, (e.g. due to the reasons described in paragraph d), then the client can simply identify which order status changed;
- if K2 is rebooted, it is easy to update the status of new and/or modified orders on the client-side when the K2 serial number of the accepted orders are known, (so that it would be in synch with the restarted K2, see Section 4.7).

The orders are also marked by the central trading system individually. The markings differ in the cash and the derivative systems (the OrdNo field of Table 8 contains integers, while the OrdNo field of the derivative Table 27 is string-type). These identifiers may also be found on the buy or sell side of any possible transactions derived from the order.

#### 4.2.4. Specific marking/identification of entered orders

We recommend the use of the BROKERREF field in the case of orders, because it will track the order from the beginning to the relevant transaction (it will appear on the buy or sell side of the trades resulted from this order). This is how an individual identifier, which arrives from the background system, can be given to each order and the fate of the order can be checked subsequently. This creates, therefore, a connection between the order initiated by the client system and the transaction sent by the central system.

#### 4.2.5. Checking the structure and content of order entries

The orders entered by the client are checked and may be blocked at various stages (i.e. 3) of the process:

- The K2 client library functions perform a quick check (before the order is sent to the K2 server): if the order is not appropriate, then the *ifsc\_orderentry* function returns with a negative value and the error may be detected with another function.
- The K2 server checks the order in terms of content: for example, in the case of an error, it will upload the Msg field of Table 11 (IFSC User Manual) with an error message, and sets the K2 status of the order to "DENIED". The client will only be informed of this if it makes queries of the table and checks the status.
- The central trading system does not accept the order (e.g. because trading was suspended). This will be indicated in Tables 11 and 8 mentioned above.

#### 4.2.6. Trading account numbers necessary for order entries

Trading accounts are necessary for order entries. These cannot be queried through K2, therefore, we suggest that these be stored in the background system of the client. Trading account numbers are used differently in the cash and the derivative systems. In fact, in the derivative system they change on a daily basis since the BSE generates them automatically every day based on the positions received from KELER (Hungarian settlement house). Special attention should be paid to this since, in the case of an incorrect trading account number, the central system will reject the order.

Composition of the account number in the cash system:

example: 0011-0000001 (for client account), or 0011-00001-H (for house account)

- where the first number is the BSE and KELER member code, while the second number is a serial number.
- the account numbers do not change in the cash system (the brokerage firm may ask for the extension or cancellation of these).

Composition of the account number in the derivative system:

example: 7-006-I (client account), 7-888888-I (omnibus account), 7-000000-H (house account).

Trading account numbers consist of three parts which are separated by a dash:

*Part I:*

**Client's KELER number** (max. 6 digits, without zeros at the beginning). E.g. 1, 13, 235. Or

**Company's KELER number** (2 digits where the first number is always a zero). E.g. 01, 02, 03. Or

**Special client account** (the character "8" repeated 6 times, followed by the company code, and an "I") – for new clients that do not have codes yet.

*Part II:*

**KELER code of stock exchange member** (3 digits, with zeros at the beginning)

*Part III:*

**an I, H, or M**, (I-client account, H-own account, M-market maker)

- account numbers change in the derivative system on a daily basis, based on the data sent by KELER to the BSE in the evening.

#### 4.2.7. Handling of the price and yield fields upon entering orders

The documentation does not detail how to treat the price and quantity fields when orders are entered, although these are not only special but also differ in the cash and derivative systems.

In the cash system both the price, yield, and quantity fields contain integers: prices containing decimal places should simply be multiplied by as many decimals as the given security is traded with. The multiplier number will be one of the characteristics of the given security (Securities Table 5, PriceDecimals, YieldDecimals). Attention: these may unilaterally be reset by the BSE overnight! Therefore, it is expedient to read and store these upon each start of K2. It is important to pay attention to this mainly for those systems which run continuously and do not operate in the start/shutdown mode.

The structure of the price and yield fields is special in the derivative system (IFS\_FIXREAL) and consist of two parts:

- the first field is a real number (double) which contains the real value of the price or the yield itself without rounding;
- the second field gives the number of decimals.

#### 4.2.8. Handling of the securities board id (BoardID), security (SecID) and SecBoardID identifiers

The board in MMTS terminology groups the securities with common trading rules (ex. shares). But a security can belong to more than one board in the same time. One example is the case of shares as they can be traded beside to the normal main board ("FŐRÉ"), on a board with negotiated deals ("FXRV"). When an order is entered it is important to know for which board has to be entered.

Please note that the securities board identifier (BoardID) and the security identifier (SecID or SecCode) are treated differently on the side of the K2 client (due to the characteristics of MMTS):

- the securities board id (e.g. FŐRÉ) and the security ID (e.g. MOL) must be given in two separate fields upon the entry of an order– e.g. see Table 11 in the User Manual;
- when making queries of the successfully entered orders, the name of the securities board id and the security will return in one single field (SecBoardID). In the cash system, there is a zero at the beginning (zero using ASCII character 48)



(e.g. „0FÖRÉ MOL „), see Table 8 in the User Manual. In the derivative system, there is no zero at the beginning.

For further information see the documents on MMTS I and MMTS II data types.

### **4.3. Information queries from K2, types of information**

#### **4.3.1. Types of K2 client data queries**

When the K2 client performs data queries, it must basically deal with four types of tables:

- static table: it is enough to query the content of the table only once because it will not change during the day;
- dynamically growing table: the table is continually expanding with new records (and existing records do not change), e.g. the table of trading transactions;
- tables containing both static and dynamic data: e.g. the table of securities (Tables 5 and 24), where the beginning is static and the rest is dynamic (the “inside” of the table changes).
- tables which can change completely upon each change: e.g. order books.

There are several options for the K2 client to query data:

- Query of the records of a table by **physical record serial numbers** (idx): ifs\_get\_first\_idx and ifs\_get\_next\_idx functions. With this you can find, for example, what the 16<sup>th</sup> record (line) of a table is.
- Query of the records of a table with changed contents based on **change numbers** (record): ifs\_get\_first\_record and ifs\_get\_next\_record functions. K2 stores a change number for each record of each table and increases this number if the record is modified. If the client remembers which data it has received with the latest change number, then it can make queries of the next changed record with the ifs\_get\_next\_record function.
- since there are several fields in the **securities table**, the individual fields have separate change numbers in the records, and the contents of the **changed fields** may be queried based on this (i.e. this is sort of an optimisation of bandwidth utilisation) (field): ifsc\_get\_first\_field\_chg and ifsc\_get\_next\_field\_chg.
- **Making special queries of the order books**, with separate functions (orderbook and marketbyprx): first, the order books K2 is supposed to monitor (ifs\_orderbook\_conf and ifsc\_marketbyprx\_conf) must be set on the client-side, then the data of the order books may be queried with the ifsc\_get\_first\_orderbook, ifsc\_get\_next\_orderbook, ifsc\_get\_first\_marketbyprx ifsc\_get\_next\_marketbyprx functions.

#### **4.3.2. Quicker client queries**

The K2 client programs may accelerate the querying of data from the K2 server if all changes to a table (e.g. orders) are queried, until there are no further data in the given table, and the next table is only queried afterwards. For example, if you query a record from the orders first and then another one from the trades and so on. This is not as fast when you query the whole first table, and then the whole second table. This operation is derived from the fact how K2 server processes one query, because the IFSS gives the answers in a package. This means that it anticipates the next records and puts these into a buffer, and the IFSC can fulfil the next IFSC client request from its own buffer.

If the client does not query the next record from the same MMTS table (1 or 2), then it will discard the contents of the buffer.

#### 4.3.3. Return values of the data query functions

Although the documentation makes little mention of this, it is extremely important that in each case we interpret the return values of the data query functions at the K2 client-side (e.g. `ifs_get_next_record`):

- if the return value is zero, the function was successful;
- a negative value refers to an error which must be queried with the `get_last_errormsg` function because the value of the error will be overwritten at the next function call if it also returns with an error;
- if the return value is `IFS_NOMORE`, then presently there are no more data in the table;

#### 4.3.4. Change numbers by tables

As we have already mentioned, each table registered in K2 contains records and each record has its own indicator: the change number. Upon starting, it is a serial number. The highest number is always registered by K2 and this will serve as the change number of the table. When a record changes, the change number of the table increases by one, and the K2 server sets the change number of the given record to this value. When the client makes queries of a record, it will receive its change number, so next time it will know that those records will be needed whose change number is higher than the one it has.

In the case of the securities table, the individual fields have their own separate change numbers.

### 4.4. *Client review: what kind of a K2 server is it connected to?*

When connecting to the K2 server, the client has several opportunities for checking (`ifsc_connect` function):

- **mmts\_type** field: it receives information regarding whether it has connected to an MMTSI or an MMTS II K2 server. This should be checked, since the client can define the module to which it should connect with the `ifsc_create` hostname & `service_name` parameters (and these may be reset).

- **pid** field: this is the identifier of the K2 software. With this one can check whether the previously received pid value is the same as the one currently received, and if not, then the client data and the server tables should be synchronised (with a complete new download). If the two pid values agree, then it is enough to set the change numbers of the individual tables from the client-side only and continue queries from the point where the client last stopped. A changed pid value means that K2 was restarted.
- **tradeid** field: the identifier of the central trading system. Its use is the same as that of the pid.

#### **4.5. Processing of a K2 response**

The K2 client communicates with the server through messages which are based on character series. The User Manual only mentions how the client must convert data back and forth at the end of the book. We definitely recommend that before any development work, you review the sample programs (see Section 4.11: Sample programs) and use the field coder, field decoder, and field stepper auxiliary functions (see User Manual).

#### **4.6. Review of error values and error messages after each client function call**

It is extremely important and we expressly recommend that the client program should handle the return value of each ifsc function call appropriately (as we partly described in Section 4.3.3: Return values of the data query functions). It is a general rule that if the function returns a negative value, then an error has occurred and in this case the error message must also be queried.

#### **4.7. K2 server: daily booting, interruption of connection, normal shutdown at the end of the day, shutdown in case of errors**

The booting, re-start, and shutdown of K2 as well as the end of trading require special attention and planning on the client-side. The K2 server consists of two separate parts:

- the PGW maintains the connection with the central trading system
- the IFSS maintains the connection with the K2 client applications

The K2 server should be booted at the beginning of the day and it should be shut down at the end of the day, or when the trading is finished, (in the case of a longer line interruption the PGW will stop automatically). What can and should the client do in this case?

##### **4.7.1. The normal daily booting of K2**

After booting K2 in the morning, the following should be done on the client-side (inter alia):

- note the parameters of the connection (see 4.4).
- note the change numbers of the tables
- program which order books the client wants to see (*ifs\_orderbook\_conf* and *ifsc\_marketbyprx\_conf* functions).

#### 4.7.2. Restarting K2 during the day

If K2 must be restarted completely, (in the case of a longer line interruption or the failure of the central system), then the client must do the following:

- it must connect to the K2 server again
- the parameters of the connection must be checked (see 4.4)
- all data must be downloaded again completely, or the previous change numbers of the tables must be set again, and the query should continue from that point (depending on the type of reboot)
- the order books the client wants to see must be programmed again (*ifs\_orderbook\_conf* and *ifsc\_marketbyprx\_conf* functions), and the data of these must be downloaded.
- the status of previously-sent orders must be synchronised, orders with a "U" (unknown) status must be handled.

The last point is very important in terms of daily procedures: if K2 stops for any reason and a few orders remained under "U" status: it means that K2 tried to send the order to the central server before the failure but it did not succeed or did not receive an answer. In this case, K2 does not do anything to these orders upon restart, therefore, the client must decide whether it will put them into the "Confirmed" portfolio (in this case K2 will send them again to the central trading system), or the Denied portfolio (in this case K2 will not send them to the central trading system).

Obviously, there are significant extra tasks derived from rebooting.

#### 4.7.3. Normal shutdown of K2 at the end of the day

When the trading is finished, as the BSE shuts down the central trading system, the PGW process on K2 will, after a while, stop automatically. At that time, the K2 client can still query the K2 data, but, of course, no more orders can be entered. The client has the opportunity to inquire whether the PGW is operating normally, which means it acquires information on whether there is a connection with the central trading system.

The client can get this information with the *ifsc\_get\_systime* function. The *K2QueryTimeOffset* field defines how many seconds ago the connection with the central system was successful. We suggest that you review the following values and share the status with the broker:

- if the value of the *K2QueryTimeOffset* field is <10 sec, then it is under normal daily operation
- if  $10 \leq K2QueryTimeOffset < 180$  sec, then there is temporarily no connection with the central trading system;
- if *K2QueryTimeOffset* >180 sec, then it is very likely that the connection with the central trading system has failed permanently.

The PGWSTATE field also contains useful pieces of information. If it is false, then this means that PGW is not running any more and manual intervention is needed for the reboot or shutdown of K2.

#### ***4.8. Multiple K2 client processes and thread programming***

When we were getting to know the K2 software, it turned out that for programming purposes, it is expressly forbidden for a client program to create several connections with the same K2 server through multiple multithreaded programs. It means that from the data buffers used on the client side by the IFSC, only one belongs to a client application. As a result of this, it would definitely be harmful if a new thread initiated a new data query when one thread could not complete its own data query, since in this case the data area used by the first thread will be overwritten. Therefore, individual data queries must be finished completely before the client starts a new query.

An application can create several connections to the K2 server, but this should not be done through thread programming but rather through a different cycle. Several applications which run parallel to one another can also create separate connections to the K2 server.

#### ***4.9. Recommended daily cycle for communication with K2***

From the above information, it is obvious that maintaining the connection with the K2 server from the client-side is not entirely trivial, especially if there is an error or when the server is rebooted. The flow chart illustrating the recommended process is at the end of this document.

#### ***4.10. Tracing on the client's side***

If any doubts arise regarding the kind of data sent by the client to the K2 server, the client-side development may be facilitated if client-side tracing is switched on temporarily using the `ifs_set_trace` function. With this function, the K2 client will store the messages sent to the server as well as the responses received in a text format in an `ifstrace.log` file. The developers can provide information on the structure of the file.

#### ***4.11. Sample programs, useful utilities***

Development can be facilitated by reviewing the sample programs attached to the system; these can also be found in a C source code:

- `demo.c`: ifsc (k2 client) sample program
- `gen_order`, `get_ob`, `get_table_ob_watch`, `order_status`: sample programs for entering orders, making queries of order books, and approving orders.

The **rdump1** and **rdump2** programs are of special importance. These can be started from the command line, and with their help the change in status of the orders sent from the client-side can be checked on the server. This can be useful in development work. Of course, the programs can be found on the server in the bin library, while the log files to be searched for are located in the log library.

#### ***4.12. Paying attention to new securities and changes to other trading parameters***

During the development of the K2 client, special attention should be paid to the fact that the central system is almost completely supplied with parameters and that these parameters may change. For example, the names of the securities may change along with the number of the decimal places traded in the price, new securities may come in, the duration of trading in a security may be prolonged, etc. In this case, the client should be sure to use the data which are downloadable from K2 to the greatest extent possible, as opposed to the data available in its own system. Furthermore, the client should also update these data upon each booting/new connection.

### **5. Other information on the daily operation of K2**

#### ***5.1. Booting K2 in the morning and its shutdown in the evening***

Earlier, we mentioned that according to the logic of K2, the K2 server must be booted in the morning and shut down in the evening. When trading is finished, the PGW process will stop automatically. At this time, the client programs are still in contact with K2 (the connection is not interrupted) and data may be queried. The IFSC processes must be stopped manually on the server with the k2stop command. In this case, of course, client communication with the K2 server will be interrupted.

#### ***5.2. Manual setting of K2 client user data***

The login name and password that can be used on the client-side are stored by the server in an isfc.uaf file. This file can only be extended and modified manually, since the process has not yet been automated. Any modification is read by K2 only at the next booting.

#### ***5.3. Periodical setting of the time with K2***

K2 will put a time stamp on orders received from the client-side. As the K2 server will not reset the system time of the hardware automatically, it is expedient to set the time of the K2 hardware before booting the K2 software manually.

