Chapter 6. Working with other programs

6.1 Data export via DDE server 2
6.2 Exporting data using ODBC..... 6
6.3 Exporting instruments via ODBC 10
6.4 Points to be aware of when exporting data 11
6.5 Export of data into technical analysis systems using built-in tools..... 11
6.6 Configuring wealth-lab developer 15
6.7 Configuring AmiBroker 20
6.8 Export of data into technical analysis systems using external programs 23
6.9 Transaction import 25
6.10 Importing transactions via API 39
6.11 APPENDIX 59

This chapter describes online interaction of the QUIK system with other programs, how to export information into other Windows applications (MS Excel, ODBC, Wealth-Lab Developer, AmiBroker), and how to import transactions from a file.

6.1 Data export via DDE server

button 

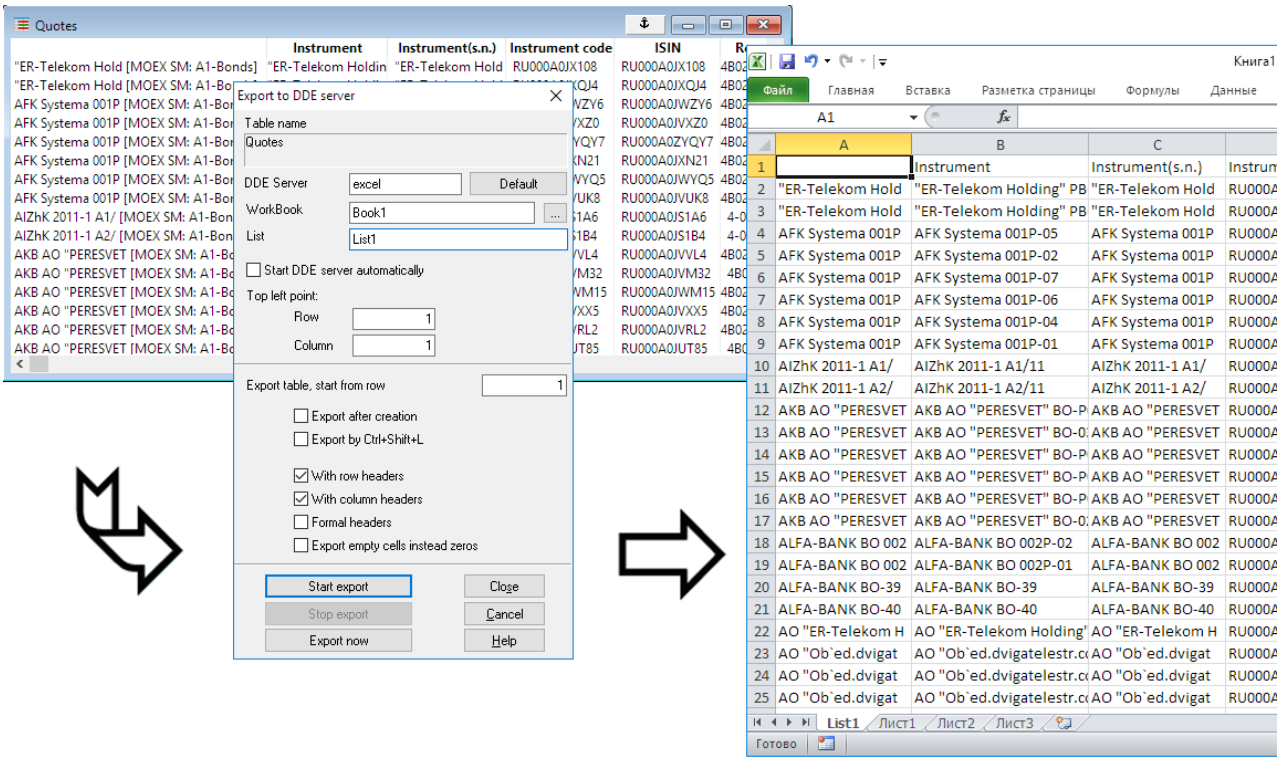
6.1.1 Purpose

DDE export allows the user to export data from QUIK tables into external programs, such as Microsoft Excel. Data is exported in the XLTABLE format using DDE method.

> **Please note that if any filters are applied to the tables, data from the tables in which you can insert or remove lines into the beginning or middle of the table (for example, cash positions and positions in instruments tables) will not be exported correctly via a DDE server.**

6.1.2 Using data export

1. Before configuring data export, a DDE server must be configured. To export data into MS Excel, open the program and load the file into which the data will be transferred (if the **Start DDE server automatically** option is disabled).



2. To export data from an active table, use one of the following methods:
   - click  on the toolbar;
   - select **Action / Export to DDE server...**;
   - hit Ctrl+L;
   - select **Export to DDE server...** from the context menu.
3. The **Export to DDE server...** configuration window contains the following options:

Option	Description
Table name	The QUIK table that serves as a data source
DDE server	The name of the DDE server. The **Default** button sets **excel** to this field
Workbook	The name of the file into which the information from the table will be exported. Click the **'...'** button on the right to select an existing file on your drive. If the **excel** is selected as a DDE server, the workbook file should have the .xls or .xlsx extension
Worksheet	The name of the worksheet into which information from the table will be transferred
Launch DDE server application automatically	Enable auto-opening of the selected workbook in MS Excel when export is started. This checkbox is enabled if **excel** is selected as a DDE server. The **Workbook** and **Worksheet** fields are optional. If the check box is disabled, then specify the name of the downloaded xls or xlsx file without path and quotes as the **Workbook** parameter. If the file already existed and was opened, specify the extension. If the file has not yet been saved, the extension is not specified
Upper left corner: _ Row; _ Column	The coordinates of the initial cell for data transfer: _ The row number, counting from the top; _ The column number, counting from the left
Export beginning from row	Export the table rows starting from the specified number. This option can be used to reduce the time of subsequent export of tables, for example, the **Time and Sales** table
Export after creation	Select this option to activate export immediately after the table configuration file is loaded, for example, after QUIK is started
Export by pressing Ctrl+Shift+L	When this checkbox is selected, you can start the export from a table by pressing this shortcut keys
With row headers	Output row headers of a QUIK table as the first column
With column headers	Output column headers of a QUIK table as the first row
Formal headers	Output the system (service) names of the headers. This feature can be useful for programming
Output blank cells instead of zeros	Output the cells that contain zeros as blank (not filled with numbers). This feature is useful for graphic display of data in MS Excel to prevent 'gaps' in chart lines where data is absent (not received from the server)

4. Click **Start export** to start exporting data from a table. If the **Launch DDE server application automatically** checkbox is selected, when this button is clicked, the program will attempt to launch MS Excel and load the specified workbook and worksheet. If MS Excel is not launched, the **Workbook** field must contain the name of the workbook with a full path to an existing file or must be empty.
   The names of workbooks in MS Excel that were created but not yet saved must be specified without extension, for example, Workook1. If the workbook name is specified without extension, the program will try to find this book among those already open in MS Excel, and,

if the book is not found, create a new book. The name of that new book might be different from the name specified in the **Workbook** field.
If no book name is specified, MS Excel will create a new book when launched. If no worksheet name is specified, the program first searches for the worksheet with the name of the table being exported (for example 'Order table #2') and if it is not found, a new worksheet with this name will be added. If other tables are exported into MS Excel and the workbook name is not specified, the same workbook will be used.
If a file name without full path is specified in the **Workbook** field, the program will search for that file in the current work directory. The system displays an error message if the file is not found.

5. You can stop export of data from a table by clicking **Stop export**.
6. The **Export now** button is intended for one-time data export.
7. Click **Close** to save configured settings and close the export configuration window. If the export process is started, it will be performed automatically in the background.
8. Click **Cancel** to close the window without saving the settings.
9. Click **Help** to open the help window for this feature.

> 1. **When any settings window is opened in MS Excel, the data is stored in the DDE buffer instead of being transferred to Excel tables. If Excel is busy for a long period, data transfer might be interrupted. DDE connection timeout (if MS Excel is busy) can be specified in the export configuration.**
> 2. **Each table has its own data export settings; therefore, different tables can export data to different DDE servers, files, MS Excel worksheets or cells.**
> 3. **Data from a QUIK table can be exported only to one DDE server.**
> 4. **However, one DDE server can accept exported data from several QUIK tables. For example, you can export data from different QUIK tables into different cells of a single MS Excel worksheet.**
> 5. **If table format is changed, the system stops export and opens the export configuration window so that the user can set new parameters for data export.**
> 6. **Using filters in tables to be exported is not recommended when exporting via DDE. If the configuration of a table has the Show zero values option, enabling this option is recommended.**
> 7. **DDE export of tables is performed according to the default sorting settings; current sorting settings are ignored.**

6.1.3 DDE export settings

menu **System / Settings / General settings...**, section **Program / Data export**

Settings 1-3 are used to configure the handling of timeouts if the DDE server is busy.

1. Time during which data receipt confirmation is pending:

   - **If the whole table is exported (1-3600)** sets a timeout for initial data transfer. The recommended value is 20;

- **If the next row is exported (1-600)** sets a timeout for updating data. The recommended value is 5.

2. **Number of attempts to resume if export fails** sets the number of permitted attempts to restore the connection when a transmission fails. The default value is 0.
3. **New thread for each DDE server** enables multithread data export to different DDE servers. The default setting is OFF.
4. Microsoft Excel supports localised versions of Microsoft Excel for different languages. The default language is English. If a Russian version of Microsoft Excel is available, this feature allows the use of English or Russian.

> **To avoid interruptions during data transfer when the DDE server is busy (e.g., when configuring formulas in a Microsoft Excel spreadsheet), the second parameter may be increased to 30-40 seconds.**

6.1.4 Useful hints

1. For simple data export to a DDE server you can copy a QUIK table by selecting **Make a copy** from the context menu or by pressing Ctrl+C. Then, you can just paste data into MS Excel (the **Edit / Paste** menu or Ctrl+V).
2. If the DDE server needs to be configured before processing exported data, use the **Export now** function first. The data will be exported once and become available for you to configure the settings. In this case DDE buffer will not be overflowed and the export will not be terminated. After configuration is complete, you can enable dynamic data export by clicking the **Start export** button.
3. If simultaneous export of data to a DDE server from several tables was interrupted and it is difficult to resume it manually, you can load a saved configuration of tables with export settings from a file (the **Export after creation** checkbox must be selected). Export will resume automatically.
4. QUIK lets you create any number of tables of the same type. To facilitate export of data, you can create a separate tab, e.g., 'For export', and put all data source tables into it. Thus, (1) those tables will not take space on the screen space, (2) it will be easier to find which tables are used for export, (3) if a DDE connection is terminated, it will be easier to find the tables for which export was suspended.
5. When exporting from several tables, it is recommended that the **Export by pressing Ctrl+Shift+L** checkbox be checked. This enables manual start of export by simply pressing this keyboard shortcut.
6. Export of multiple tables can be stopped by using the **Services / Data export/import / Stop DDE export from tables** command or 'Ctrl+Shift+S' keys.

6.1.5 Error messages

1. 'Failed to establish DDE connection. Excel is not running or the [Workbook][Worksheet] worksheet is not loaded.'

   - The DDE server or MS Excel is not running. Open the program and load the required file into it;

- Incorrect name of an MS Excel Workbook (file) or its worksheet;
- The file name in the **Workbook** field must exactly match the name in MS Excel. If the file has been saved earlier, it has an extension and must be specified with that extension, for example, quikexp.xls. The names of MS Excel workbooks that haven't been saved as files have no extensions (for example, Workbook1) and should be specified without extensions in the export settings;
- If the setting **Start DDE server automatically** is not selected then you should specify only name of the file in **Workbook** field, do not specify the path to this file;
- The **Ignore DDE request from other applications** checkbox is selected in the MS Excel settings (the **General** tab under **Service / Parameters**). Clear it.

2. 'Data exchange timeout: server overloaded.'

   - DDE connection is lost due to overload of the DDE server (MS Excel). If the server was busy because some configuration windows were opened for a long time, close those windows and start dynamic export from the **DDE export** window again. If disconnections occur regularly, increase the timeout for outputting a line **(**the **Program / Data export** section under **System / Settings / General settings...**), for example, to 30-40 seconds.

3. 'Failed to launch Excel'

   - The MS Excel program is not found on the computer.

4. 'Failed to open [Workbook] workbook in Excel, [Exported table] table'

   - The file with the specified name does not exist in the specified path or (if no path is specified) it is not found in the current working folder.

5. 'The [Worksheet] worksheet is not found in the [Exported table] table of the [Workbook] workbook'

   - The specified workbook does not contain a worksheet with the specified name.

6.2 Exporting data using ODBC

button 

6.2.1 Purpose

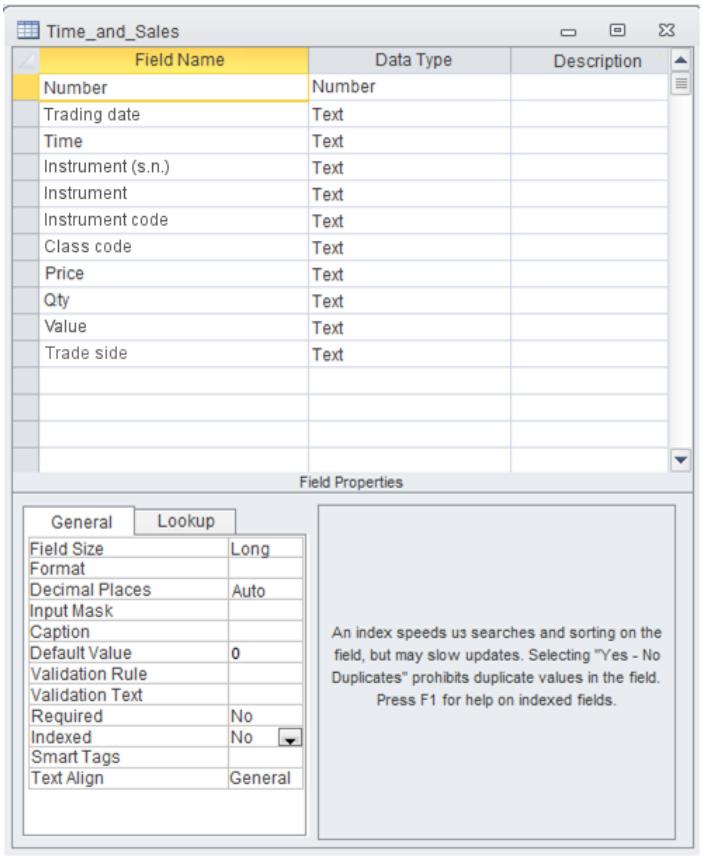
ODBC export allows the user to export data from QUIK tables to other applications for further use (storage, processing). This function can be used to directly connect the QUIK system with other programs that require online information from an exchange.

6.2.2 Using ODBC data export

> **An ODBC source should be created using the 64-bit ODBC Data Source Administrator (path: %windir%\system32\odbcad32.exe).**

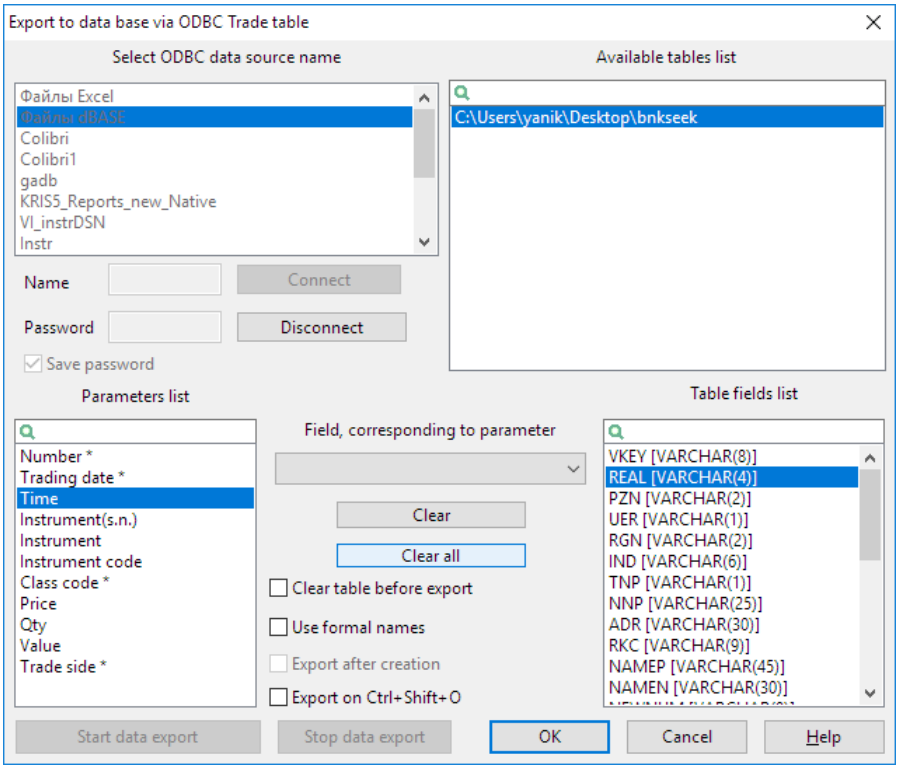
1. Before exporting data, the user should create a table that will receive this data. The structure of this table should be similar to the structure of the QUIK table from which data will be exported. In other words, its list of parameters should be the same as the list of column headers of that QUIK table. The list of table parameters and their data types are provided in

the appendix to this section. As an example, let us consider export of a trade table into a Microsoft Access table (see fig.):



2. After a table is created, it should be registered as an ODBC data source in the Windows Control Panel (select **ODBC data sources** in the **Start / Settings / Control Panel** menu).
3. In the QUIK system, you can activate the table from which data will be exported and open the ODBC export configuration window using one of the following methods:

   _ Click  on the toolbar;
   _ Select **ODBC export** from the shortcut menu;
   _ Select **Action / Export to ODBC** from the program menu;
   _ Press Ctrl+O.
4. Select a data recipient in the **Choose ODBC data source name** list. The name should be the same as described in item 2.
5. If user authorization is required for connection to the source, enter the user name and password into the corresponding field.
6. Click **Connect**. The full list of tables from the selected source will appear in the **Available tables list**.



7. Select the table to which data will be exported. The fields of the selected table will be displayed in the **Table fields list**.
8. Configure the correspondence between the fields of the QUIK table and that of the recipient table by performing the following actions for each field:

   _ Select a field of the QUIK table from the **Parameters list**;

   _ Link it with a field from the **Fields that correspond to the parameter** list. This list displays fields that are compatible with the type of the field of the QUIK table. If there is no required field, see 6.2.4 Error Messages.

> **It is not necessary to link all fields of the tables. You can do it only for the fields that contain parameters to be exported. However, the fields marked with '*' must be linked with their counterparts.**

9. If you want to remove the link between the fields, select the field of the QUIK table in the **Parameter list** and click **Clear**. If you want to apply this to all fields, click the **Clear all** button.

> **If a recipient table was selected erroneously, when you select another table from the List of available tables, the links between the fields that have the same names and data types will remain. This won't apply to fields with different names or data types.**

10. Purpose of export settings:

   _ **Clear table before export** removes old data from the table before export or when the session, sever or user is changed; if this checkbox is cleared, new data will replace the old data as it is received;

- If **Use formal names** is selected, the system identifiers of the parameters will be exported instead of their text values. For example, instead of an instrument's name and class, their codes in a trading system will be exported; enumerable types, such as 'order direction', will be replaced with the symbol codes (for example, 'B' for buying 'S' for selling);
- If the **Export after creation** checkbox is selected, data export will begin simultaneously with loading the window configuration into the QUIK system, for example, when the program is launched;
- When the **Export by Ctrl+Shift+O** checkbox is selected, you can activate export from a table by pressing this keyboard shortcut.

11. Click the **Start data export** button to start export.

> **When exporting data from the Quotes Table, make sure that the Current and historical data option is selected in the data reception configuration (the system / Settings / General settings... menu, then the Program / Saving data section).**

12. Click **Stop data export** to stop data export.

13. Click **OK** to close the export configuration window saving the changes. If the export process is started, it will be performed automatically in the background.

14. To close the export configuration window without saving any changes, click **Cancel**.

6.2.3 Useful hints

1. Do not use reserved words such as 'money', 'group', 'order', 'number', 'date', etc, to name table fields.
2. Do not use spaces in table names, table fields, and paths to tables.
3. To export data from QUIK tables that contain parameters with the '*' symbol, enable the **Clear table before export** checkbox. This will remove all data related to previous server connection sessions.
4. If you need to archive information received from QUIK in a database, we recommend creating two tables with the same structure. Use one of them to store data from QUIK and the other as an archive. Copy data to the archive at the end of a trading session or at the beginning of the next one. You can thus avoid saving the data twice, for example, in case when you need to load data from the QUIK system again.
5. Data from one QUIK table can be exported only to one recipient table. However, you can configure several identical tables in QUIK to be exported into different applications.
6. If data export is used regularly, check the **Export after creation** box in the export settings. When it is enabled, the export procedure will start automatically when the program is launched.
7. If several tables are used for data export, for convenience, we recommend creating a separate tab and putting the tables in it. This would keep data export configuration from possible changes in a similar table used to view market information.
8. When exporting from several tables, it is recommended that the **Export by pressing Ctrl+Shift+O** checkbox be selected. This enables manual start of export by simply pressing this keyboard shortcut.

9. If the tables in the database are indexed using a key, to ensure uniqueness, this field should be configured to correspond to the **Order number** and **Class code** fields in QUIK.
10. If the configuration of a table has the **Show zero values** option, enabling this option is recommended.

6.2.4 Error messages

1. 'Syntax error during INSERT INTO operation.'
   The most common error caused by ignoring hints 1 and 2.
2. 'Connection error. Not enough information to connect to DSN.'
   The ODBC data source is not configured or incorrectly configured. See 6.2.2.
3. 'The data recipient is not specified in the data source list.'

   - The table is not registered in the list of ODBC data sources. See 6.2.2, item 2;
   - A field from the **Table field list** does not have a corresponding field in the **Fields that correspond to the parameter** list;
   - A field in the recipient table is incompatible with the type of the QUIK table field. The field type in the recipient table must match its corresponding field in the QUIK table.

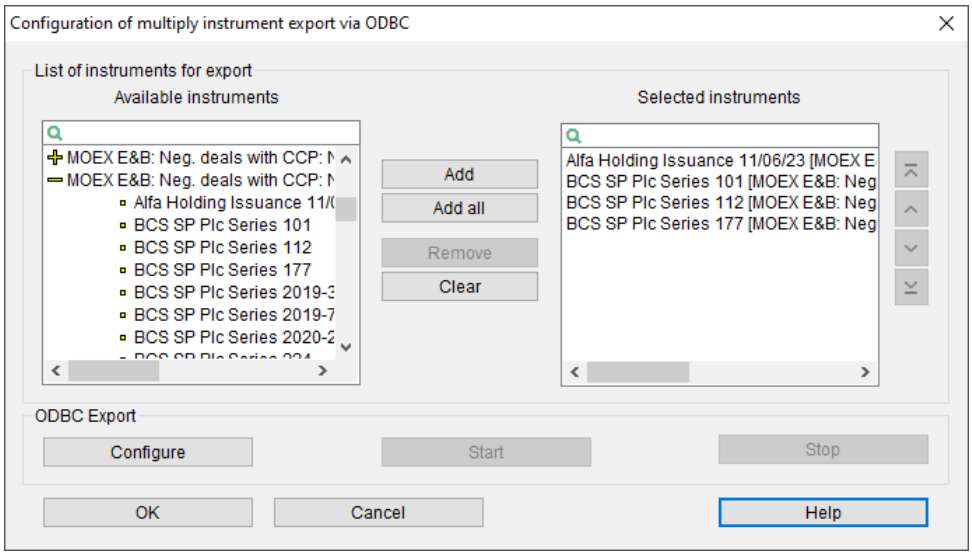
6.3 Exporting instruments via ODBC

menu **Services / Data export /import/ ODBC export instruments...**

6.3.1 Purpose

Exporting orders queues for instruments (DOM) through ODBC without creating **Level II Quotes** tables for them.

6.3.2 Using instruments export



1. Configure a list of instruments to be exported by generating the **Selected instruments** list from available instruments.
2. Click the **Configure** button to configure the list of parameters to be exported. Parameters are configured as described in 6.2.2.
3. To begin export, click **Start**. To stop export, click **Stop**.

Click **Yes** to close the export window and save all parameters. Click **Cancel** to close the window without saving the configured settings.

> 1. **Sparse Level II quotes are exported using DDE in the non-sparse form.**
> 2. **When exporting, the precision of float type fields of database table should not exceed the value '6'.**

6.4 Points to be aware of when exporting data

There are several points that you should be aware of when exporting data via DDE or ODBC from the tables operating in the 'drag-and-drop' mode:

1. When columns are moved or deleted in the 'drag-and-drop' mode (**Tables / Ask for confirmation when moving and deleting items using Drag-and-Drop** under the **System / Settings / General settings...** menu) from the tables for which DDE / ODBC export is configured, the system will reload export settings and display the configuration window.
2. If user filters in a table are changed, the system reloads DDE/ODBC export configuration and displays the appropriate configuration window.
3. The user can delete or change the order of rows using 'drag-and-drop' in the **Quotes** table, the **Option parameters** table and the **Buy / Sell** table. The **Options board** table allows you only to remove rows in this mode. When these actions are performed, the system reloads export settings and opens the appropriate configuration dialogue.

6.5 Export of data into technical analysis systems using built-in tools

menu **Services / Data export/import / Technical analysis data...**

6.5.1 Purpose

This feature allows the user to use the data obtained by the QUIK system for dynamic technical analysis in business applications, such as Wealth-Lab Developer 6.x and AmiBroker 6.x.

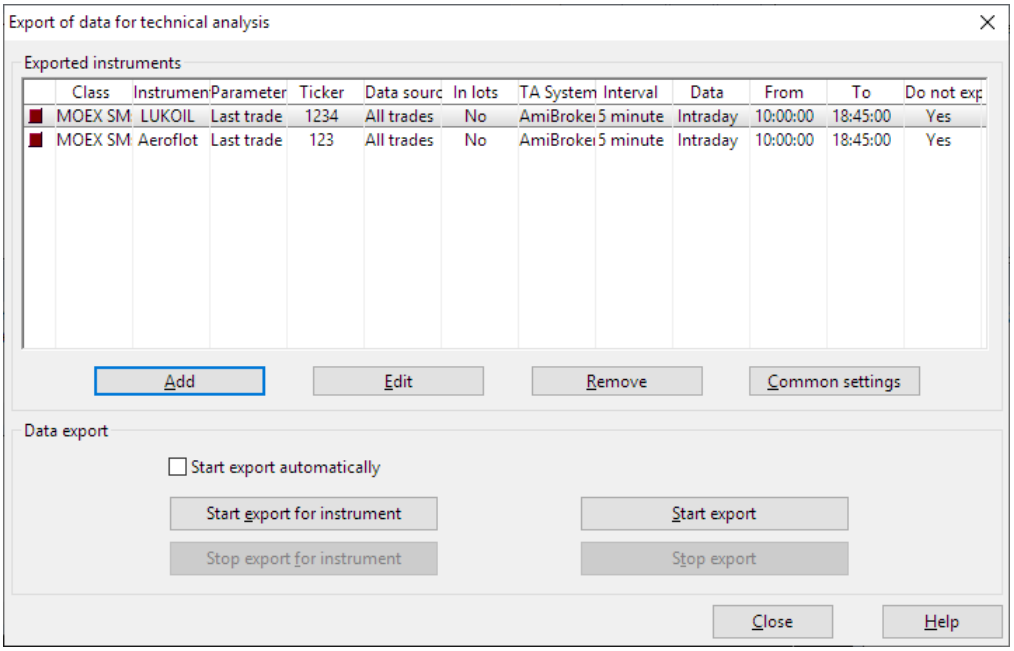
6.5.2 Using data export

You can export data using the built-in tools if the **Services / Data export/import / Technical analysis data...** menu is available.

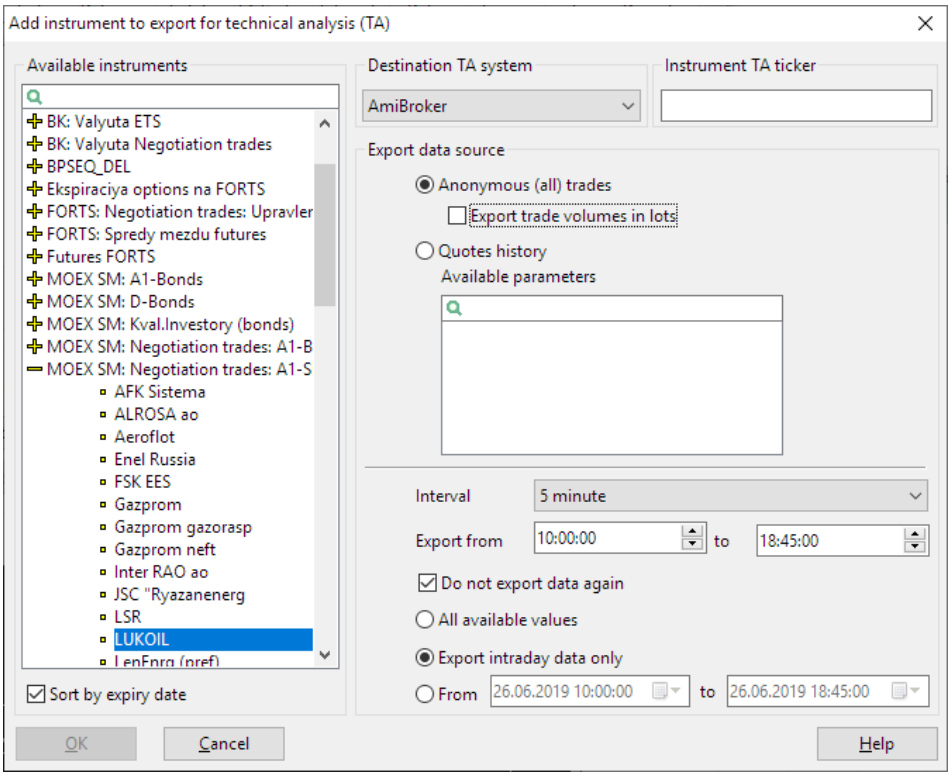
To ensure proper operation of technical analysis packages, they must be properly configures; for further details, see 6.4 – 6.7.

Also note that installing other internet trading systems that can export data to technical analysis programs can block export from the QUIK system.

1. To open the export configuration window, select **Services /Data export/import / Technical analysis data...**.



2. Create a list of instruments and their parameters to be exported. To add an instrument to that list, click **Add**. The export parameters selection dialog box will open.



3. Select the desired instrument from the list of **available instruments**. By default, the list is sorted in alphabetical order. To make selection of instruments with limited period of circulation easier, select the **Sort by expiry date** check box.
4. In the **DestinationTA system** box, select the program into which the data will be exported.
5. In the **Instrument TA ticker** box, enter the same instrument name that you specified in the data source settings for the technical analysis system.
6. Select a data source for export. From the **Anonymous (all) trades** you can export prices and volumes for each trade. If the **Export trade volumes in lots** is enabled, the volume of instruments in the last trade will correspond to the number of lots in the trade; when this

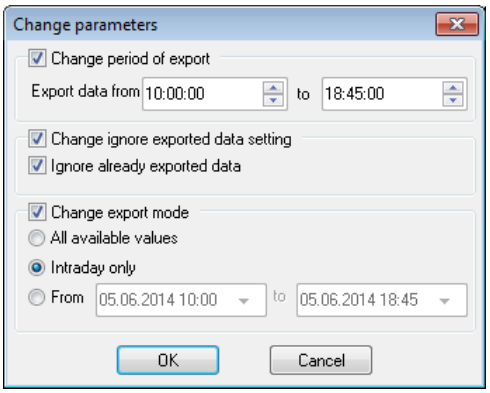
check box is cleared, the volume will correspond to the number of instruments. From the **Quotes history** table, you can export a larger number of parameters, such as **Best bid price** and **Best offer price**.

7. The **Interval** parameter determines the intervals at which the 'candlesticks' are drawn on a chart. If you select the **Tick** value, data will be exported for each trade.
8. The **Export from...** and **to...** parameters determine the beginning and end of a trading session. Edit those parameters to exclude trades in opening and closing periods when the prices might be significantly different from market averages.
9. If the **Do not export data again** check box is checked, if the server connection is lost and then restored, QUIK will not transfer data that has been already transferred. If this check box is not selected, whenever the server connection is restored, QUIK will export all data from the beginning of the current trading session. See use cases in 6.2.3 Useful hints.
10. The following parameters determine the amount of exported data:

    - Select **All available values** to export all values including historical data for previous days if the broker's QUIK server supports this function;
    - Select **Export intraday data only** to export all values in the current trading session;
    - Fill in the **From...** and **to...** boxes to export values from the specified period.

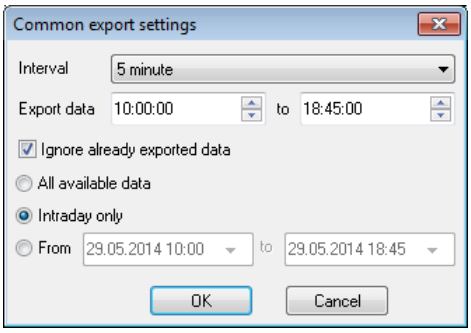
> **The parameters no. 7-10 by default are configured according to general data export settings (the Common settings button in the configuration dialog box for exporting data to technical analysis systems).**

11. Click **OK** to close the instrument parameter selection window. The new instrument will be displayed in the **Exported instruments** list.
12. To edit an instrument created earlier, click the **Edit** button.
13. To edit the common export parameters for several instruments, highlight the desired instruments in the list and click **Edit**. The dialog box will open that allows you to change export settings such as export period, repeated export of data, and the range of exported values.



14. To remove instruments from this list, highlight them and click **Remove**.
15. The **Common settings** button allows the user to specify default parameters that would be used when a new instrument is added for export. This button opens the following dialogue window:

- The **Interval** parameter determines the intervals at which the 'candlesticks' are drawn on charts for all exported instruments;
- The **Export data...** and **to...** parameters determine the beginning and end of a trading session;
- If the **Ignore already exported data** check box is selected, if the server connection is lost and then restored, QUIK will not transfer data that has been already transferred. If this box is clear, whenever the server connection is restored, QUIK will export all data from the beginning of the current trading session;



- The following parameters determine the volume of exported data (apply to all instruments in the list):

  Select **All available data** to export all values including historical data for previous days if the broker's QUIK server supports this function;

  Select **Intraday only** to export all values in the current trading session;

  Fill in the **From... to...** boxes to export only the parameters for the specified interval. For example, if you do not want to export data on trades executed during the first minute of a trading session, you can configure transfer of data not from the very beginning of trading, but from the specified time.

16. If **Start export automatically** is checked, data export will begin as soon as the program launches.
17. Click the **Start export for instrument** button to start export procedure for a selected instrument. The instruments that are being exported are marked with the ▶ symbol in the list.
18. Click **Start export** to start export procedure for all instruments in the list.
19. The **Stop export for instrument** button stops export for a selected instrument, while **Stop export** stops export procedure for all instruments in the list.

> **After export of data for an instrument starts, that instrument cannot be removed from the list and its settings cannot be changed. If you need to make any changes, first stop data transfer, then edit the list of instruments to be exported.**

20. Click the **Close** button to close the export configuration window. If the export process is started, it will be performed automatically in the background.

6.5.3 Useful hints

1. Check the **Ignore already exported data** box in the QUIK export configuration. If the server connection was lost and then restored, data export will be resumed automatically. Only newly received data will be transferred.

2. If the QUIK workstation is closed by the user, data export stops. When the system is launched again, data is exported from the beginning of the current trading session.

6.6 Configuring wealth-lab developer

6.6.1 Getting started

1. Install the Wealth-Lab Developer program.

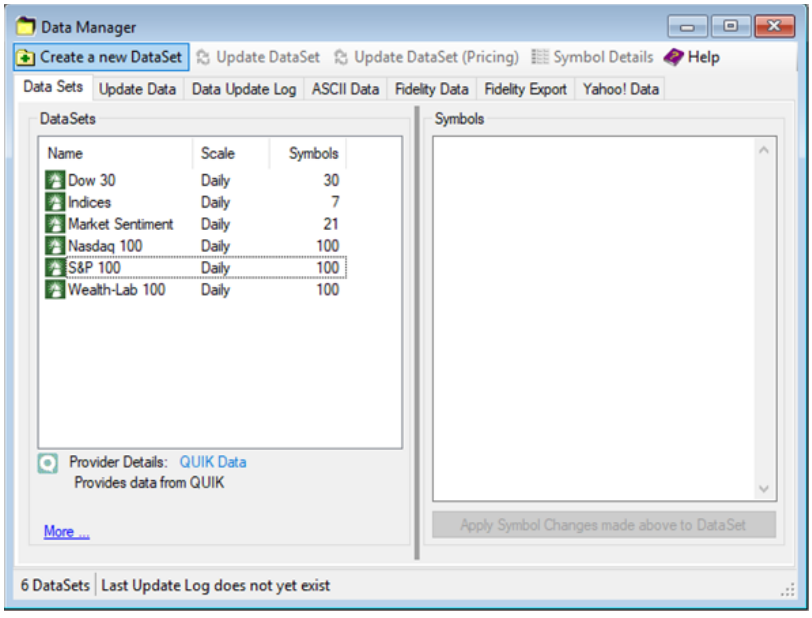
   - Download the installation package for the provider that performs export from QUIK into Wealth-Lab Developer. It can be downloaded from: http://arqatech.com/upload/iblock/644/quik2wld_rtadapter.zip.

2. To export data from QUIK to the Wealth-Lab Developer program, the WealthLab.DataProviders provider is required. This provider supports the following operation modes: one-time data reception and real-time reception. To use the provider in both modes, install the provider as follows:

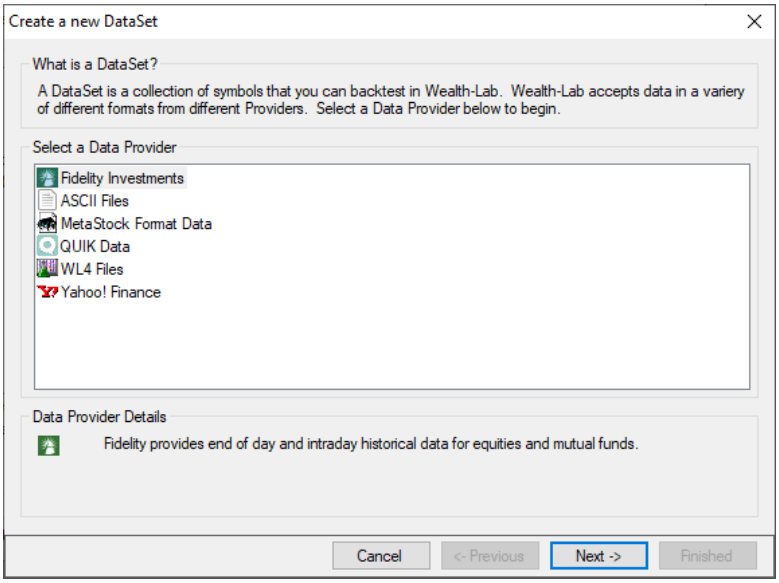
   - Unpack the received archive and copy the following files to the directory where the Wealth-Lab Developer program is installed;
     - **WealthLab.DataProviders.QUIK.dll**;
     - **NLog.dll** and **NLog.xml**, if they are not present in the Wealth-Lab Developer distribution kit or the **NLog.dll** file has older version (the **NLog.dll** version can be checked in the file properties);
   - Run the Wealth-Lab Developer application through the **Start** menu.

3. Configure the settings of the Wealth-Lab Developer application:

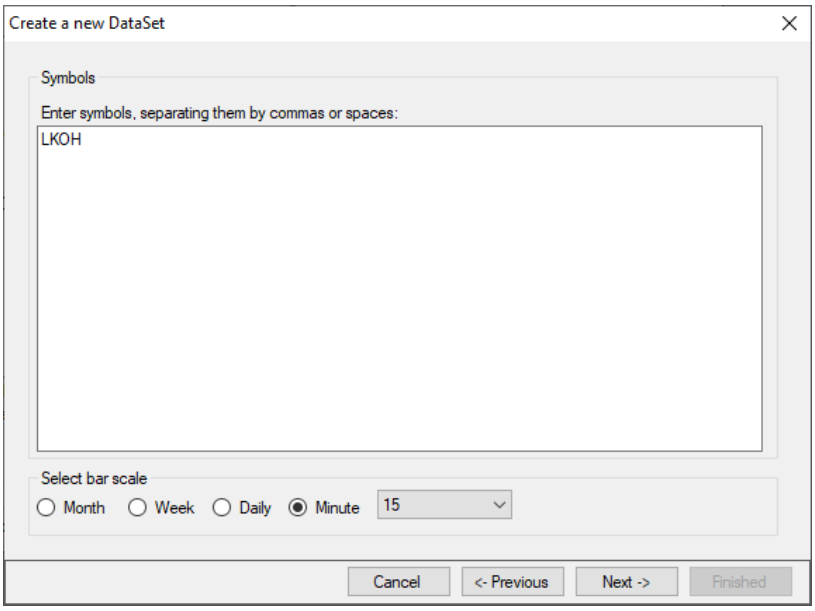
   - On the main menu of the application, select **Tools / Data manager**;
   - In the Data Manager window, select the **Create a new DataSet** command:



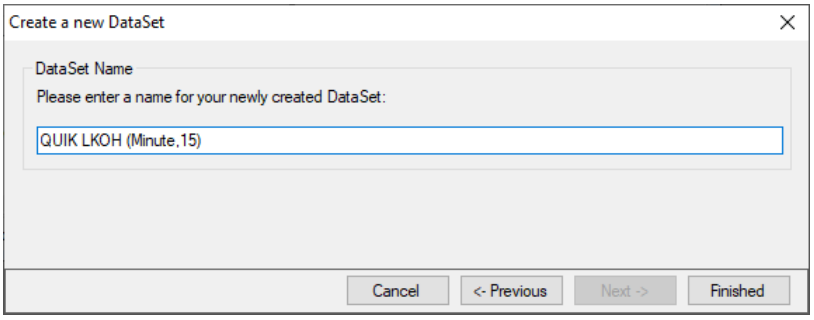
   - In the dialog box that opens select the list of available providers **QUIK Data** and click **Next**:



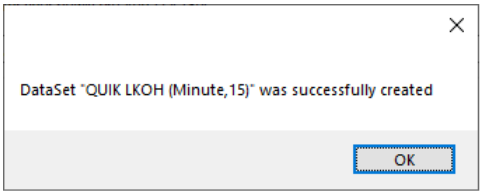
_ On the **Symbols** list, enter instrument tickers (separated by commas or spaces) which are expected to work with, select the time period (Month, Week, Daily, Minute) and then click **Next**:



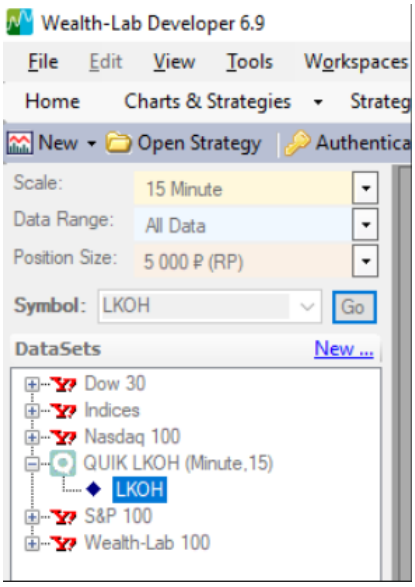
_ Enter the name of the data source for export (for example, QUIK LKOH (Minute,15)) and click **Finished**:



If the data addition was successful, the result will be shown in the Data Manager start window.

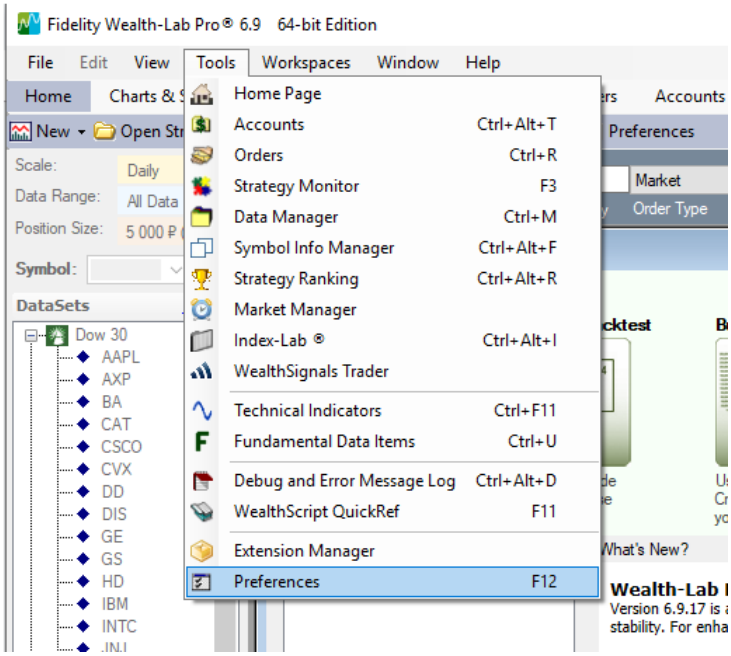


- The created DataSet will appear in the DataSets tree of the Wealth-Lab Developer program:

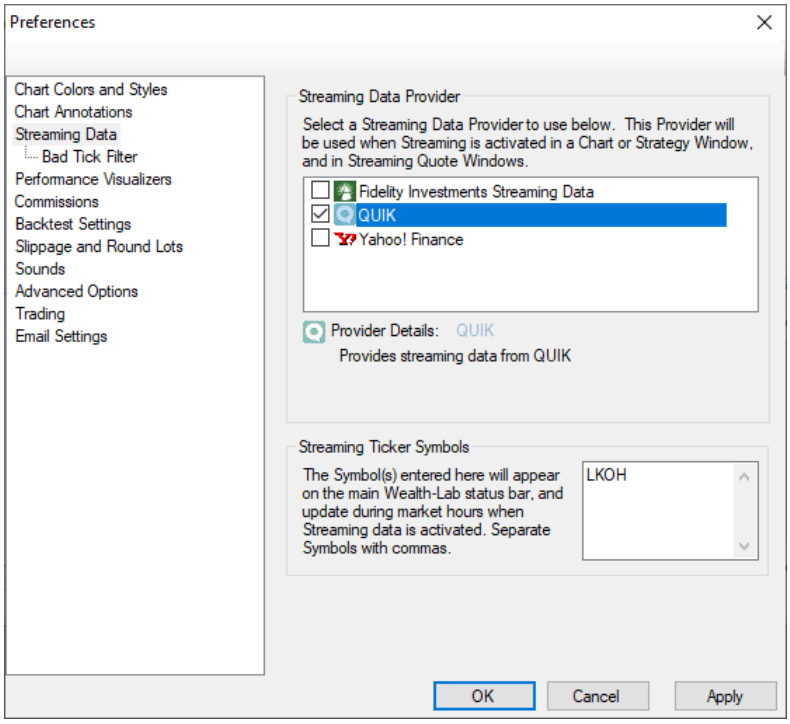


After these settings are complete, the user may receive data from the QUIK system in one-time mode.

- To receive real-time data, set additional settings:
  - Select **Tools / Preferences** on the main menu:



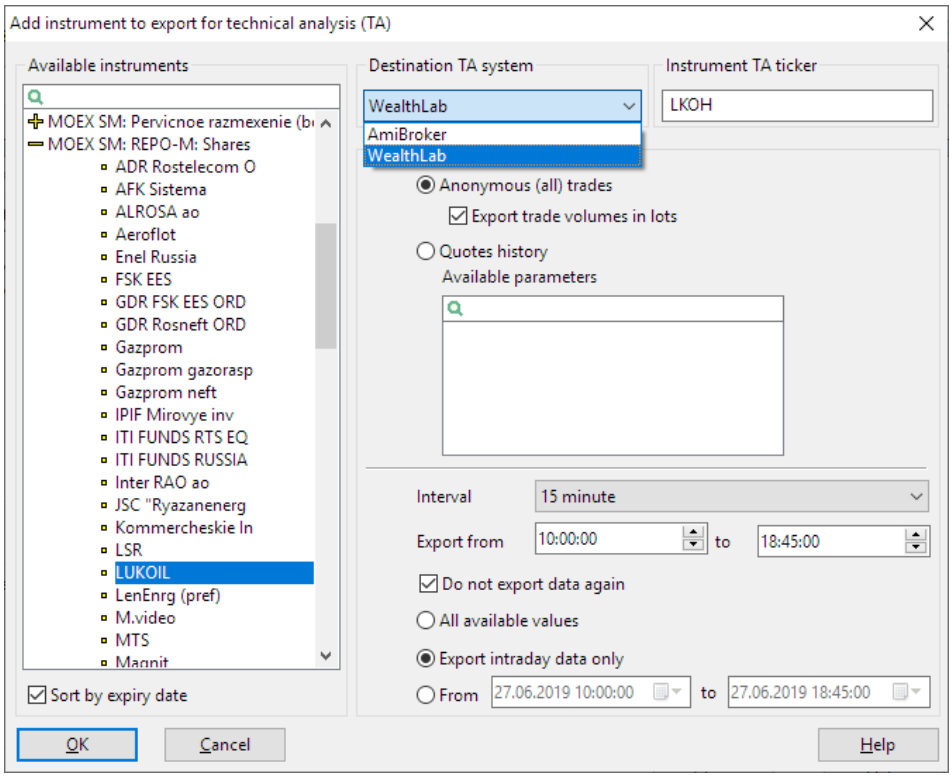
– In the **Preferences** dialog box that opens select **Streaming data** in the tree on the left and then select **QUIK** in the list of available providers on the right. Add tickers required for real-time update (for example, LKOH):



– After starting the data export (see 6.5) click the **Stream** button in the bottom right corner of the Wealth-Lab Developer program. After that the Wealth-Lab Developer program will start receiving real-time data from the QUIK system:



6.6.2 Configuring a data source in QUIK



1. Open the export configuration window in QUIK (select **Services / Data export/import / Technical analysis data...**).
2. Create a list of instruments and their parameters to be exported. To add an instrument to that list, click **Add**. The export parameters selection dialog box will open.
3. Select the desired instrument from the list of **available instruments**, for example, 'LUKOIL'.

> **An instrument you select must match an instrument you have added in the Wealth-Lab Developer application (for the application settings, see 6.6.1).**

   By default, the list is sorted in alphabetical order. To sort instruments with limited period of circulation, use the **Sort by expiry date** option.
4. Select **WealthLab** in the **Destination TA system** box.
5. Enter an instrument identifier in Wealth-Lab Developer into the **Instrument TA ticker** box.
6. Select a data source for export. From the **Anonymous (all) trades** table you can export prices and volumes for each trade. If the **Export trade volumes in lots** is enabled, the volume of instruments in the last trade will correspond to the number of lots in the trade; when this checkbox is cleared, the volume will correspond to the number of instruments. From the **Quotes history** table, you can export a larger number of parameters, such as **Best bid price** and **Best offer price**.
7. In the **Interval** box select an interval at which data will be exported into Wealth-Lab Developer.
8. Select the beginning and the end time of a trading session.
9. If the **Do not export data again** check box is checked, if the server connection is lost and then restored, QUIK will not transfer data that has been already transferred. If this box is

clear, whenever the server connection is restored, QUIK will export all data from the beginning of the current trading session.

10. Set parameters that determine the volume of exported data (for the selected instrument).
11. Click **OK** to close the instrument parameter selection window. The new instrument will be displayed in the **Exported instruments** list.

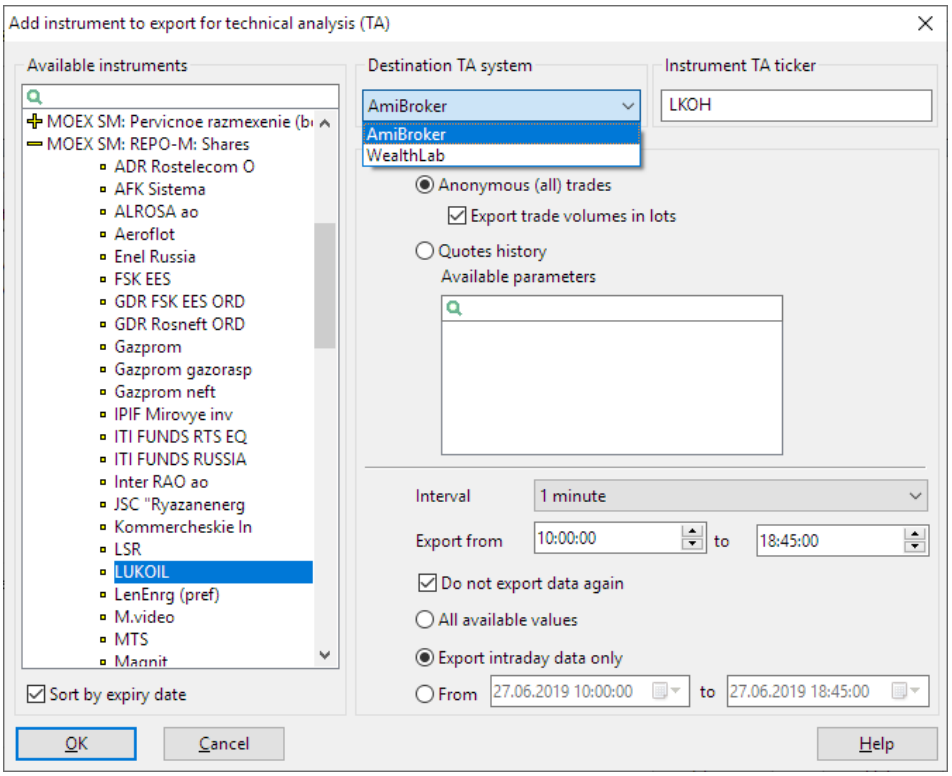
6.7 Configuring AmiBroker

6.7.1 Getting started

1. Install AmiBroker.
2. To set up export from QUIK into AmiBroker you will need QUIK2AMIBROKER_DataPlugin.dll. It can be downloaded from the official website of QUIK.
3. Copy the QUIK2AMIBROKER_DataPlugin.dll file into the **Plugins** folder in the AmiBroker directory.

6.7.2 Configuring a data source in QUIK

1. Open the export configuration window in QUIK (select **Services / Data export / Technical analysis data...**).
2. Create a list of instruments and their parameters to be exported. To add an instrument to that list, click **Add**. The export parameters selection dialog box will open.

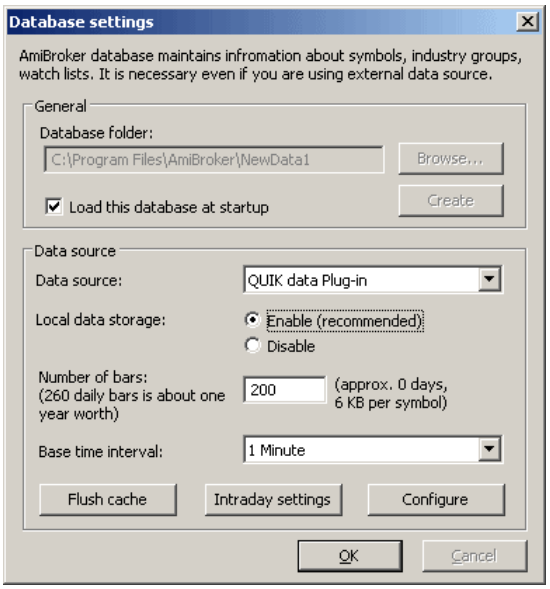


3. Select the desired instrument from the list of **available instruments**, for example, 'LUKOIL'. By default, the list is sorted in alphabetical order. To sort instruments with limited period of circulation, use the **Sort by expiry date** option.
4. Select **AmiBroker** in the **Destination TA system** box.
5. Enter an instrument identifier in AmiBroker in the **Instrument TA ticker** box.
6. Select a data source for export (**Anonymous (all) trades** table or **Quotes history** table).

7. In the **Interval** box, select an interval at which data will be exported into AmiBroker.
8. Select the beginning and the end time of a trading session.
9. If the **Do not export data again** check box is checked, if the server connection is lost and then restored, QUIK will not transfer data that has been already transferred. If this check box is clear, whenever the server connection is restored, QUIK will export all data from the beginning of the current trading session.
10. Set parameters that determine the volume of exported data (for the selected instrument).
11. Click **OK** to close the instrument parameter selection window. The new instrument will be displayed in the **Exported instruments** list.

6.7.3 Configuring data import in AmiBroker

1. Launch AmiBroker and select **Database...** under **File / New.**
2. In the **Database folder** box in the window that opens, specify the path to the database in which the imported information will be stored. If there is no database, you need to create it. To do this, specify the path and unique name and click **Create**. Select **Load this database at startup** to load the specified database upon program launch.

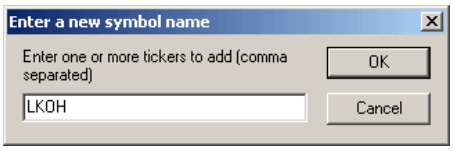


3. Then configure data import settings. Select **QUIK Data Plug-in** as a data source in the **Data source** box.
4. The **Local Data Storage** parameter must be set to **Enable** (the default setting).
5. Specify the number of bars to be displayed in a chart in the **Number of bars** box.
6. Specify the data import interval in the **Base time interval** box.

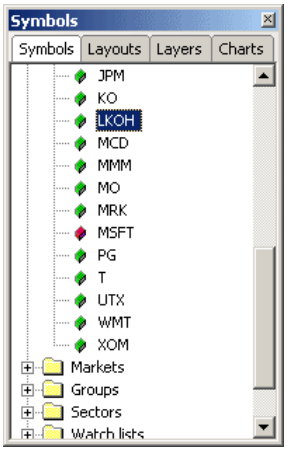
> 1. **The value of the Base time interval parameter must be the same as the value of the Interval box in QUIK export settings for AmiBroker (see 6.7.2), otherwise data can't be exported.**
> 2. **AmiBroker allows the user to change chart intervals through the context menu (select Intraday in the menu) based on the global chart settings in the Database settings. After changing a chart's timeframe configuration, the chart in Amibroker is usually different from the chart exported from QUIK due to specific way of generating chart intervals in the QUIK system. QUIK divides an hour into intervals specific for the chart (for example, into 15-minute**

intervals) and displays the first chart bar in the next closest interval. AmiBroker plots charts in a different way, so the charts may differ.

7. Click **OK** to save your settings.
8. After the settings are saved, the **Configure** button will become available. Press this button to open the **Settings for data export from Quik to AmiBroker** window. Set the **Clear AmiBroker data after establishing connection** check box to remove all data transferred to AmiBroker before starting a new export session from QUIK. If this check box is not set, new data will be added to the previous data, i.e., AmiBroker will store quotation history.
9. Then you should create a new ticker. Select **New** under the **Symbol** menu. In the window that opens, enter the same ticker name that you specified when configuring the data source in the QUIK system (see 6.7.2). In this example, enter 'LKOH' and press Enter.



10. In the **Symbols** window you can see the list of all created tickers. When you select a ticker, the chart that corresponds to that ticker will be displayed in the AmiBroker's workspace.



11. To start export, click the **Start export for instrument** button in the **Export of data for technical analysis** window in the QUIK workstation. The received data will be displayed in real time in the AmiBroker's workspace during the process.



6.8 Export of data into technical analysis systems using external programs

6.8.1 Purpose

This function allows the user to perform dynamic technical analysis of the data gathered by QUIK using Equis MetaStock in case the built-in export tools are unavailable.

6.8.2 Usage

There are two common ways to export information: using the DDE2MS or MetaServer RT. Both programs transfer information as follows:

- Data is transferred from QUIK's **Quotes** table into Microsoft Excel via DDE;
- DDE2MS or MetaServer RT obtains continuously updated data from Microsoft Excel and transfers it to MetaStock.

QUIK → Microsoft Excel → DDE2MS or MetaServerRT → MetaStock

In our opinion, DDE2MS is more convenient to configure and work with, but, unlike MetaServer RT, whose demo version can download data for two instruments, DDE2MS can only work with one. You can choose either of these options.

6.8.3 Using DDE2MS

1. Launch the DDE2MS program (it comes as a portable .exe file that does not need installation).
2. Specify parameters required for operation:

   - In the **Path** box, specify the full path to the MetaStock data source you created (for example, C:\MetaStock Data\SBER);
   - In the **Symbol** box specify the data source name ('SBER' in this example);
   - Select '1' in the **Interval** box;
   - Enter 'Excel' in the **Server** box;
   - In the **Topic** box, specify which Worksheet and Workbook data will be taken from: in the format '[book_name.xls]sheet_name', for example: '[Workook1.xls]Worksheet1';
   - In the **Item-Last Price** (last trade price) and **Item-Volume** (number of instruments in the last trade) boxes, enter the addresses of the MS Excel cells from which data will be taken, in the 'RxCy' format, where 'x' is a row number, 'y' is a column number.

3. Click **Start**. If DDE2MS can find data to be processed, a table with the current OHLCV values will appear, the button name will change to **Stop** and the input boxes will be disabled. Otherwise an error message will be displayed.

6.8.4 Using MetaServer RT

1. Launch MetaServerRT.
2. Check whether the source created in MetaStock ('SBER' in this example) is present in the list of available sources. If it isn't, select **Symbol-Add new** from the menu and enter the description of the created source (SBER).
3. Select the table row that corresponds to the new data source (contains its name in the 'TS / MS Symbol' column) and double click it to open the source configuration window.
4. In that window perform the following steps:

   - Enter the data source name in the **Global Server / MetaStock Symbol** box;
   - Enter 'Excel' in the **DDE** box.

5. Go to the Easy-Setup tools tab:

   - In the **Topic** box specify the Worksheet and Workbook containing QUIK data, in the format '[book_name.xls]sheet_name', for example: '[Workook1.xls]Worksheet1';
   - In the **Item** box specify the address of the cell that contains data, in the 'RxCy' format, where 'x' is a row number, 'y' is a column number: 'R1C1'.

6. Click the **Set** buttons.
7. Go back to the **Tradestation/MetaStock** tab and check all boxes. Click **OK**.
8. Click the green button (that has the **Start all instruments** tip) on the MetaServerRT program's toolbar. If the settings were configured correctly and the instrument data is available, the '*' parameter should change to '+' in the table row where the source is configured.

6.8.5 Configuration

> **In order MetaStock ba able to correctly handle fractional numbers, set '.' instead of ',' as the decimal separator in Windows' Regional Settings, otherwise data will not be transferred correctly.**

6.9 Transaction import

6.9.1 Universal format of transaction import file

Universal format allows to send transactions if the description is known.

Descriprtion of fields see in the **Create transactions pocket** window Chapter 3, "Viewing Information", sub-section 3.50.4.

Select **Save to tri file** from the context menu of the order in the **Transaction pocket** table.

Format of .tri file
This file is a sequence of lines each of which contains information about a single transaction.

Line forming principle:

1. TRANS_ID, CLASSCODE, ACTION lines, the ACTION value is the name of required transaction, for example: 'Repo with CCP order entry'.
2. Transaction parameters are specified in the following format: 'PARAMETER_NAME=parameter_value' separated by ';'.

Examples of lines that might be contained in this file:

Transaction	Line
An order for REPO with CCP 1 day trade Buying 1000000.00 at 6.1500	TRANS_ID=1; CLASSCODE=EQRP_BND; ACTION=New order (CCP); Trade account=L01+00000F00; Order operation=Buy; Order type=Limit; Price spliting flag=At different prices; Fill or kill=Put in queue; Price enter as...=By price; Is market-maker=By default; Security board=EQRP_BND; Instrument=RU000A0JRF37; Price=6.1500; Quantity=0; Comment=4F161/T2/; Order value=1000000.00; External user id=; Add. info=;
An iceberg order In the MOEX stock market, buy 100 lots of Aeroflot at 70 each, the visible number of lots in the queue is 10, the client code is 467.	CLASSCODE=TQBR; TRANS_ID=2; ACTION=Place Iceberg order; Trading account=S01-00000F00; B/S=Buying;Type=Limit;Type by price=by different prices; Type by balance=place in queue; Price value entry type=By price; Instrument=AFLT; Price=70; Lots=100; Visible number=10; Note=467;

Transaction	Line
An iceberg algo order to buy 99 AGZD at the price 111.0000, limit, visible quantity – 2, cancel algo order upon cancelling a linked order, aggressive – normal, execution condition – GTC	TRANS_ID=4;CLASSCODE=ALGO_ICEBERG;ACTION=New Order;Trading account=S01-00000F00;Firm=NC0038900000;Side=Buy;Type=Limit;Class=BQUOTE;Instrument= AGZD;Price=111.00;Quantity=99;Displayed Quantity=2;Client code=Q3;Comment=;Kill by Linked Order=Y;Aggressive=Normal;Time in force=Good Till Cancel;Expiry Date=20161125;Expiry Time=124141;Enable Execution Alerts=Yes;
A VWAP algo order to buy 5 AGZD, Catch Up percent – 0.000000, Tick Up – 0	TRANS_ID=2;CLASSCODE=ALGO_VWAP;ACTION=New Order;Trading account=L01-00000F00;Firm=NC0038900000;Side=Buy;Class=BQUOTE;Instrument= AGZD;Qty/Value=Qty;Value=0.00;Quantity=5;Need market percent=No;Value percent=0;Start Time=183001;Stop Time=183501;Till Close Session=No;Time Type=Current; Iteration set mode=Iterations Count;Iterations Count=5;Iteration Time=0;Deviation from VWAP=0.000000;Client code=Q3;Comment=;Minimum Price=0.000000;Maximum Price=0.000000;Enable Execution Alerts=No;CatchUp-order=Yes;Catch Up Percent=0.000000;Tick Up=0;Use IOC-orders=Yes;Use Limit Inside Iterations=Yes;Work in price range=No;Use time profile=No;Include own trades=Yes;
A TWAP algo order to buy 222 AGZD, at price 150.0000	TRANS_ID=1;CLASSCODE=ALGO_TWAP;ACTION=New Order;Trading account=S01-00000F00;Firm=NC0038900000;Side=Buy;Class=BQUOTE;Instrument= AGZD;Qty/Value=Qty;Value=0.00;Quantity=222;Type=Limit;Start Time=183003;Stop Time=183003;Time Type=Current; Iteration set mode=Iterations Count;Iterations Count=10;Iteration Time=15;Price Limit Type=Percent;Price Limit (percents)=0.000000;Price Limit (money)=0.00;Client code=Q3;Comment=;Enable Execution Alerts=No;Use time profile=No;Price=150.0000;
A GTD algo order to buy 10 LKOH at price 1000.00	TRANS_ID=3;CLASSCODE=ALGO_GTD;ACTION=New Order;Trading account=S01-00000F00;Firm=NC0038900000;Side=Buy;Class=BQUOTE;Instrument=LKOH;Price=1000.00;Quantity=10;Type=Limit;Client code=Q3;Comment=;Time in force=Good Till Cancel;Expiry Date=20160912;Expiry Time=162335;Enable Execution Alerts=No;Use work interval=No;Start Time=0;Stop Time=0;Counterparty=NC0038900000;Settlement code=T0;
Cancel an algo order with identifier 000000000072ALGO_TWAP	TRANS_ID=3;CLASSCODE=ALGO_TWAP;ACTION=Canel an order;Order Identifier=000000000072ALGO_TWAP

6.9.2 Specific format of transaction import file

menu **Services / Data export/import.../ Import transactions from file...**

Purpose

This feature is used to automatically send transactions prepared by an external program to the trading system and receive reports about transaction import.

Usage

QUIK exchanges data with external programs via text files that have specific format:

- *.tri is a file with transaction parameters;
- *.tro is a file that contains the results of sending transactions into a trading system;
- *.trr is a file that contains the log of transaction processing.

> **Tri files and tro files are not recommended to be placed online.**

The pattern of interaction between the programs is as follows:

1. An external program creates a transaction with specified parameters and writes it as a new line into a .tri file. Transactions are identified by an additional unique integer parameter TRANS_ID.
2. The QUIK system reads the .tri file periodically at certain intervals and transfers unprocessed transactions to the trading system. If the description of a transaction does not match the accepted format, the transaction is rejected.
3. The result of these actions is written into a .tro file in the format that can be recognized by the external program. Each line in that file contains information about the processing of a transaction identified by its TRANS_ID parameter.
4. The external program reads the results of the operations (on MOEX) in the .tro file. The registration number of an order is specified in the text message (the 'DESCRIPTION' field) received from the trading system and in the ORDER_NUMBER parameter.

> **Before QUIK reads the .tri file for the first time, it reads the processed orders from the .tro file. The orders contained in the .tro file are considered processed, and the lines in the .tri file with the same TRANS_ID will be ignored. If the external program upon launch enumerates the orders from the beginning, remove the .tro file from the working directory before launching the program.**

Format of .tri file

This file is a sequence of lines each of which contains information about a single transaction. Transaction parameters are specified in the following format:
'PARAMETER_NAME=parameter_value' separated by ';'.

> **An Iceberg order transaction on MOEX should be specified in a special format, which can be found in the example below.**

Parameters and their values:

Parameter	Description
CLASSCODE	The class code for the transaction, for example, TQBR. Required parameter
SECCODE	The ID of the instrument in the transaction, for example, SBER

Parameter	Description
ACTION	The transaction type. Valid values: _ NEW_ORDER refers to a new order; _ NEW_NEG_DEAL refers to a new OTC order; _ NEW_REPO_NEG_DEAL refers to a new REPO order; _ NEW_EXT_REPO_NEG_DEAL refers to a new modified REPO (REPO-M) order; _ NEW_STOP_ORDER refers to a new stop order; _ KILL_ORDER cancels an order; _ KILL_NEG_DEAL cancels an OTC or REPO order; _ KILL_STOP_ORDER cancels a stop order; _ KILL_ALL_ORDERS withdraws all orders from the trading system; _ KILL_ALL_STOP_ORDERS – cancels all stop orders; _ KILL_ALL_NEG_DEALS - withdraws all orders for OTC and REPO trades; _ KILL_ALL_FUTURES_ORDERS cancels all orders on the Moscow Exchange derivatives market; _ MOVE_ORDERS moves orders on the Moscow Exchange derivatives market; _ NEW_QUOTE refers to a new non-addressed order; _ KILL_QUOTE cancels a non-addressed order; _ NEW_REPORT refers to a new order-report for transaction confirmation in the NDM and REPO modes; _ SET_FUT_LIMIT refers to a new restriction on a futures account. Required parameter
FIRM_ID	The trader's identifier (firm ID)
ACCOUNT	The trader's account number. The parameter is required if 'ACTION' = 'KILL_ALL_FUTURES_ORDERS'. This parameter is case sensitive (upper / lower case characters)
CLIENT_CODE	A 20-characters compound field, which can contain a client code and a text comment (the **Comment** field) with the same separator that is used to enter orders manually. Optional parameter
TYPE	The type of order, an optional parameter. Valid values: 'L' refers to limited (by default), 'M' refers to market
MARKET_MAKER_ORDER	Indicates that the order is a Market maker's order. Valid values are 'YES' or 'NO'. The default value (if the parameter is not specified) is 'NO'
OPERATION	The direction of an order, a required parameter. Valid values: 'S' refers to selling, 'B' refers to buying
EXECUTION_CONDITION	The execution condition of an order, an optional parameter. Valid values: _ PUT_IN_QUEUE refers to moving an order to the queue (default); _ FILL_OR_KILL refers to immediate execution or cancelling; _ KILL_BALANCE refers to withdrawal of balance
QUANTITY	The number of lots in an order, a required parameter
REPOVALUE	The value of a REPO-M trade in roubles
START_DISCOUNT	The initial discount value in an REPO-M order
LOWER_DISCOUNT	The minimum discount value in an REPO-M order

Parameter	Description
UPPER_DISCOUNT	The maximum discount value in an REPO-M order
PRICE	The order price, per instrument unit. Required parameter. When placing a market order (TYPE=M) to the Moscow Exchange derivatives market, specify the worst possible price (minimum or maximum, depending on direction). The order will be filled at the market price anyway. For other markets, specify price=0. For take profit type orders, the parameter is not used
STOPPRICE	Stop price, per instrument unit. Used only when 'ACTION' = 'NEW_STOP_ORDER'
STOP_ORDER_KIND	The type of a stop order. Valid values: _ SIMPLE_STOP_ORDER refers to a stop-limit order; _ CONDITION_PRICE_BY_OTHER_SEC refers to a conditional order with a condition on another instrument; _ WITH_LINKED_LIMIT_ORDER refers to an order with a linked order; _ TAKE_PROFIT_STOP_ORDER refers to a take-profit order; _ TAKE_PROFIT_AND_STOP_LIMIT_ORDER refers to a take-profit and stop-limit order; _ ACTIVATED_BY_ORDER_SIMPLE_STOP_ORDER refers to an 'If done' stop-limit order; _ ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER refers to an 'If done' take-profit order; _ ACTIVATED_BY_ORDER_TAKE_PROFIT_AND_STOP_LIMIT_ORDER refers to an 'if done' take-profit and stop-limit order. If this parameter is missing, the order is considered to be a stop-limit order
STOPPRICE_CLASSCODE	The class of the instrument used in the condition. Used only for 'STOP_ORDER_KIND' = 'CONDITION_PRICE_BY_OTHER_SEC'
STOPPRICE_SECCODE	The code of the instrument used in the condition. Used only for 'STOP_ORDER_KIND' = 'CONDITION_PRICE_BY_OTHER_SEC'
STOPPRICE_CONDITION	The direction of the stop price condition. Used only for 'STOP_ORDER_KIND' = 'CONDITION_PRICE_BY_OTHER_SEC'. Valid values: '<=' or '>='
LINKED_ORDER_PRICE	The price of a linked limit order. Used only for 'STOP_ORDER_KIND' = 'WITH_LINKED_LIMIT_ORDER'
EXPIRY_DATE	Expiration date of a stop order. Valid values: _ GTC: before cancellation; _ TODAY: before the today's trading session ends; _ Date format: YYYYMMDD
STOPPRICE2	The 'stop-limit' condition price for a take-profit and stop-limit order
MARKET_STOP_LIMIT	The flag to indicate whether the order should be filled at the market price when the 'stop-limit' condition is met. Valid values are 'YES' or 'NO'. Applies to take-profit and stop-limit orders
MARKET_TAKE_PROFIT	The flag to indicate whether the order should be filled at the market price when the 'take-profit' condition is met. Valid values are 'YES' or 'NO'. Applies to take-profit stop-limit orders
IS_ACTIVE_IN_TIME	The flag that indicates that a take-profit and stop-limit order will be valid during a certain period of time. Valid values are 'YES' or 'NO'

Parameter	Description
ACTIVE_FROM_TIME	Time when an order of the 'Take-profit and stop-limit' type becomes valid, in the format HHMMSS
ACTIVE_TO_TIME	Time until which an order of the 'Take-profit and stop-limit' type remains valid, in the format HHMMSS
PARTNER	The code of the partner involved in an OTC transaction. Used for 'ACTION' = 'NEW_NEG_DEAL', 'ACTION' = 'NEW_REPO_NEG_DEAL' or 'ACTION' = 'NEW_EXT_REPO_NEG_DEAL'
ORDER_KEY	The number of an order to be cancelled. Used for 'ACTION' = 'KILL_ORDER' or 'ACTION' = 'KILL_NEG_DEAL' or 'ACTION' = 'KILL_QUOTE'
STOP_ORDER_KEY	The number of a stop order to be cancelled. Used for 'ACTION' = 'KILL_STOP_ORDER'
TRANS_ID	The id of an order with values from 1 to 18 446 744 073 709 551 615
SETTLE_CODE	A settlement code for OTC orders
PRICE2	A price of the second leg of a REPO transaction
REPOTERM	REPO period. This parameter relates to REPO-M trades
REPORATE	REPO rate as a percentage
BLOCK_SECURITIES	The flag that indicates that instruments should be blocked during a REPO operation ('YES', 'NO')
REFUNDRATE	Fixed rate of refunding in case the 2nd part of REPO is not executed, as a percentage
COMMENT	A text comment specified in an order. Used when a group of orders is cancelled
KILL_IF_LINKED_ORDER_ PARTLY_FILLED	The flag that indicates that a stop order should be cancelled after its linked limit order is partially executed. Used only for 'STOP_ORDER_KIND' = 'WITH_LINKED_LIMIT_ORDER'. Valid values are 'YES' or 'NO'
OFFSET	The offset of the last trade price from the maximum (minimum). Used for STOP_ORDER_KIND = TAKE_PROFIT_STOP_ORDER or ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER
OFFSET_UNITS	Offset measurement units. Valid values: _ PERCENTS refers to percentage (with steps of 1/100th of percent); _ PRICE_UNITS indicates that offset is measured in terms of price (with steps equal to price increments for this instrument) Used for STOP_ORDER_KIND = TAKE_PROFIT_STOP_ORDER or ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER
SPREAD	Protective spread size. Used for STOP_ORDER_KIND = TAKE_PROFIT_STOP_ORDER or ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER

Parameter	Description
SPREAD_UNITS	Protective spread measurement units. Valid values: _ PERCENTS refers to percentage (with steps of 1/100th of percent); _ PRICE_UNITS indicates that offset is measured in terms of price (with steps equal to price increments for this instrument) Used for STOP_ORDER_KIND = TAKE_PROFIT_STOP_ORDER or ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER
BASE_ORDER_KEY	The registration number of the primary order. Used for STOP_ORDER_KIND = ACTIVATED_BY_ORDER_SIMPLE_STOP_ORDER or ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER
USE_BASE_ORDER_ BALANCE	This flag specifies that the volume of an 'if done' order should be equal to the number of instruments executed in the primary order. Valid values are: 'YES' or 'NO'. Used for 'STOP_ORDER_KIND' = 'ACTIVATED_BY_ORDER_SIMPLE_STOP_ORDER' or 'ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER'
ACTIVATE_IF_BASE_ ORDER_PARTLY_FILLED	This parameter specifies that the 'if done' order should be activated when its primary order is partially filled. Valid values are: 'YES' or 'NO'. Used for 'STOP_ORDER_KIND' = 'ACTIVATED_BY_ORDER_SIMPLE_STOP_ORDER' or 'ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER'
BASE_CONTRACT	Identifier of the base contract for futures and options. This parameter is required for cancelling orders on the Moscow Exchange derivatives market
MODE	The mode of moving orders on the Moscow Exchange derivatives market. This parameter relates to the action 'ACTION' = 'MOVE_ORDERS'. Valid values: _ 0 – do not change orders' volume; _ 1 – change orders' volumes to new values; _ 2 – if new and old values do not match in at least one order, both orders are canceled; _ 3 – the quantity of orders to be submitted is determined as a difference between the new value and the executed quantity of the corresponding order. If new quantity is less then the executed one in at least one order, both orders are canceled
FIRST_ORDER_NUMBER	The first order's number
FIRST_ORDER_NEW_ QUANTITY	The first order's volume
FIRST_ORDER_NEW_ PRICE	The first order's price
SECOND_ORDER_ NUMBER	The second order's number
SECOND_ORDER_NEW_ QUANTITY	The second order's quantity
SECOND_ORDER_NEW_ PRICE	The second order's price

Parameter	Description
KILL_ACTIVE_ORDERS	This attribute specifies that active orders for this instrument should be cancelled. Used only for 'ACTION' = 'NEW_QUOTE'. Valid values are: 'YES' or 'NO'
NEG_TRADE_OPERATION	The operation's direction in the trade confirmed by a report
NEG_TRADE_NUMBER	The number of the trade to be executed, confirmed by a report
VOLUMEMN	The limit of open positions, for 'Limit type' = 'Funds' or 'Total'
KGO	Client's collateral coefficient
USE_KGO	This parameter determines whether to load the KGO value when the limits are loaded from a file: _ for USE_KGO=Y, the KGO value is loaded; _ for USE_KGO=N, the KGO value is not loaded When you place a limit in the MOEX derivatives market with mandatory reduction (see Chapter 7, "Broker Operations", sub-section 7.7.2), you need to specify USE_KGO=Y
CHECK_LIMITS	This parameter specifies whether the order's price will be checked for being within the acceptable price range. This parameter is used for the Moscow Exchange derivatives market. This is an optional parameter for transactions that place new orders in the 'FORTS Options' and 'OTC: FORTS Options' classes. Valid values are: 'YES' performs verification, 'NO' does not perform verification
MATCHREF	A reference that links two REPO or NDM transactions. Counterparties can enter into the transaction only if they specified the same value of this parameter in their orders. This parameter is an arbitrary sequence (both numbers and letters are allowed up to 10 characters). Optional parameter
CORRECTION	The limit correction mode for futures account. Valid values: _ Y: enabled, setting a limit changes the current value; _ N: disabled (default), specifying a limit sets a new value

The commands for conditional cancellation of groups of orders ('KILL_ALL_ORDERS', 'KILL_ALL_STOP_ORDERS', 'KILL_ALL_NEG_DEALS', 'KILL_ALL_FUTURES_ORDERS') are processed as follows:

1. The 'CLASSCODE', 'TRANS_ID', 'ACTION', ACCOUNT parameters are required.
2. Possible optional parameters for commands for conditional cancellation of groups of orders:

   - 'KILL_ALL_ORDERS': 'SECCODE', 'ACCOUNT', 'OPERATION', 'CLIENT_CODE', 'COMMENT';
   - 'KILL_ALL_STOP_ORDERS': 'SECCODE', 'ACCOUNT', 'OPERATION', 'CLIENT_ CODE', 'COMMENT', 'EXPIRY_DATE';
   - 'KILL_ALL_NEG_DEALS': 'SECCODE', 'ACCOUNT', 'OPERATION', 'CLIENT_ CODE', 'COMMENT', 'PARTNER', 'SETTLE_CODE';
   - 'KILL_ALL_FUTURES_ORDERS': 'ACCOUNT', 'OPERATION', 'BASE_CONTRACT'.

3. The orders that match all parameters (logical 'AND') specified in the transaction will be cancelled.

Moving orders on the Moscow Exchange derivatives market is performed as follows:

- If MODE=0, then the FIRST_ORDER_NUMBER and SECOND_ORDER_NUMBER orders are cancelled. Two new orders are sent into the trading system. Their prices will be changed while their volume will remain the same;
- If MODE=1, then the FIRST_ORDER_NUMBER and SECOND_ORDER_NUMBER orders are cancelled. Two new orders are sent into the trading system. Both the price and volume of the orders will be changed;
- If MODE=2, then the FIRST_ORDER_NUMBER and SECOND_ORDER_NUMBER orders are cancelled. If the number of instruments in both of the cancelled orders equals the values specified after FIRST_ORDER_NEW_QUANTITY and SECOND_ORDER_NEW_QUANTITY, two new orders with the corresponding parameters are sent into the trading system.

Examples of lines that might be contained in this file:

Transaction	Line
A sell order Rostelecom, limit order, 3 lots for 43.21 roubles each.	ACCOUNT=NL0080000043; CLIENT_CODE=467; TYPE=L; TRANS_ID=1; CLASSCODE=TQBR; SECCODE=RU0008943394; ACTION=NEW_ORDER; OPERATION=S; PRICE=43,21; QUANTITY=3;
A buy order LUKOIL, limit order, 3 lots for 253.3 roubles each.	ACCOUNT=NL0080000043; CLIENT_CODE=467; TYPE=L; TRANS_ID=2; CLASSCODE=TQBR; SECCODE=LKOH; ACTION=NEW_ORDER; OPERATION=B; PRICE=253,3; QUANTITY=3;
A buy order Rushydro, market order, 15 lots.	ACCOUNT=NL0080000043; CLIENT_CODE=467; TYPE=M; TRANS_ID=7; CLASSCODE=TQBR; SECCODE=HYDR; ACTION=NEW_ORDER; OPERATION=B; PRICE=0; QUANTITY=15;
A sell order LKOH-3.10 futures, market, 15 contracts.	ACCOUNT=SPBFUT00009; CLIENT_CODE= SPBFUT00009; TYPE=M; TRANS_ID=8; CLASSCODE=SPBFUT; SECCODE=LKH0; ACTION=NEW_ORDER; OPERATION=S; PRICE=16231; QUANTITY=15;
An OTC buy order Rostelecom, limit order, 1 lot for 42.81 roubles.	ACCOUNT=NL0080000043; CLIENT_CODE=467; TYPE=L; TRANS_ID=3; CLASSCODE=PSEQ; SECCODE=RU0008943394; ACTION= NEW_NEG_DEAL; OPERATION=B; PRICE=42,81; QUANTITY=1; PARTNER=NC0080000000;
An OTC sell order RusHydro, limit order, 3 lots for 1.113 roubles each.	ACCOUNT=NL0080000043; CLIENT_CODE=467; TYPE=L; TRANS_ID=4; CLASSCODE=PSEQ; SECCODE=HYDR; ACTION=NEW_NEG_DEAL; OPERATION=S; PRICE=1,113; QUANTITY=3; PARTNER=NC0080100000;
An order for REPO trade Gasprom, selling 10 lots for 100 roubles each, within 4 days, with R90 settlements.	ACTION=NEW_REPO_NEG_DEAL; TRANS_ID=135; CLASSCODE=RPMA; SECCODE=GAZP; ACCOUNT=NL0080000043; CLIENT_CODE=E1//NOTE; PARTNER=NC0038900000; OPERATION=S; QUANTITY=10; PRICE=100; SETTLE_CODE=R90; REPOTERM=4; REPORATE=5; REFUNDRATE=6;

Transaction	Line
An order for REPO-M trade Lukoil, buying 10 lots for 16,000 roubles total, within 1 day, with S0 settlements. To execute the trade, the counterparty must place a counter-order specifying 'link' in the Reference field	ACTION=NEW_EXT_REPO_NEG_DEAL; TRANS_ID=19; CLASSCODE=RPMA; SECCODE=LKOH; ACCOUNT=NL0080000043; CLIENT_CODE=Q7//NOTE; PARTNER=NC0080100000; OPERATION=B; QUANTITY=10; REPOVALUE=16000; SETTLE_CODE=S0; REPOTERM=1; REPORATE=0; REFUNDRATE=0; BLOCK_SECURITIES=NO; MATCHREF=link
A stop-limit RusHydro, 100 lots for selling at 7.000, the stop price is 7.300, the expiration date is 05.19.2011.	ACTION=NEW_STOP_ORDER; ACCOUNT= NL0080000043; TRANS_ID=17; CLASSCODE=TQBR; SECCODE=HYDR; OPERATION=S; QUANTITY=100; CLIENT_CODE=467; STOPPRICE=7.3; PRICE=7.0; EXPIRY_DATE=20110519;
A stop order with a condition on another instrument, Rostelecom -ao, selling 15 lots at 7.000, the condition is based on Rostelecom -ap, the stop price condition is <= 8.000.	ACTION=NEW_STOP_ORDER; STOP_ORDER_KIND=CONDITION_PRICE_BY_OTHER_SEC; ACCOUNT= NL0080000043; QUANTITY=15; TRANS_ID=15; CLASSCODE=TQBR; SECCODE=RTKM; STOPPRICE_CLASSCODE=TQBR; STOPPRICE_SECCODE=RTKMP; STOPPRICE_CONDITION=<=; OPERATION=S; CLIENT_CODE=1001; STOPPRICE=8.0; PRICE=7.0;
A stop order with a linked order RusHydro, buying 15 lots for 8.500 each, stop price >= 8.000, and limit order at 6.000.	ACTION=NEW_STOP_ORDER; STOP_ORDER_KIND=WITH_LINKED_LIMIT_ORDER; ACCOUNT= NL0080000043; TRANS_ID=16; CLASSCODE=TQBR; SECCODE=HYDR; OPERATION=B; QUANTITY=15; CLIENT_CODE=1001; STOPPRICE=8.0; PRICE=8.5; LINKED_ORDER_PRICE=6.0; KILL_IF_LINKED_ORDER_PARTLY_FILLED=NO;
Take-profit Lukoil, buying 1 lot, activation when the price reaches 265, with an offset of 5% and protective interval of 5 pips, the expiration date is 07/06/2010.	ACTION=NEW_STOP_ORDER; TRANS_ID=8; STOP_ORDER_KIND=TAKE_PROFIT_STOP_ORDER; STOPPRICE=265; CLIENT_CODE=Q5; OPERATION=B; SECCODE=LKOH; CLASSCODE=TQBR; ACCOUNT=L01-00000F00; QUANTITY=1; EXPIRY_DATE=20100706; OFFSET=5; OFFSET_UNITS=PERCENTS; SPREAD=5; SPREAD_UNITS=PRICE_UNITS;
Take-profit and stop-limit Lukoil, buying 1 lot, take-profit is activated when the price reaches 2000, with an offset of 5% and protective spread of 3%, stop price is 2222, limit order price is 2255, valid from 10:00:01 to 19:45:45.	ACTION=NEW_STOP_ORDER; TRANS_ID=10055; CLASSCODE= TQBR; SECCODE=LKOH; ACCOUNT=L01-00000F00; CLIENT_CODE=Q7; OPERATION=B; QUANTITY=1; PRICE=2255; STOPPRICE=2000; STOP_ORDER_KIND=TAKE_PROFIT_AND_STOP_LIMIT_ORDER; OFFSET=5; OFFSET_UNITS=PERCENTS; SPREAD=3; SPREAD_UNITS=PERCENTS; MARKET_TAKE_PROFIT=NO; STOPPRICE2=2222; IS_ACTIVE_IN_TIME=YES; ACTIVE_FROM_TIME=100001; ACTIVE_TO_TIME=194545; MARKET_STOP_LIMIT=NO

Transaction	Line
Take-profit when an order is executed when order 81874488 is partially filled, place a take-profit order for Lukoil in the amount of the filled part of the primary order; when the price reaches 265, with an offset of 10 pips and the protective interval of 10 pips.	ACTION=NEW_STOP_ORDER; TRANS_ID=11; STOP_ORDER_KIND=ACTIVATED_BY_ORDER_TAKE_PROFIT_STOP_ORDER ; BASE_ORDER_KEY=81874488; USE_BASE_ORDER_BALANCE=yes; ACTIVATE_IF_BASE_ORDER_PARTLY_FILLED=yes; SPREAD=10; OFFSET=10; OFFSET_UNITS=PRICE_UNITS; SPREAD_UNITS=PRICE_UNITS; STOPPRICE=265; CLIENT_CODE=Q5; OPERATION=B; SECCODE=LKOH; CLASSCODE=TQBR; ACCOUNT=L01-00000F00;
Stop limit when an order is filled when order 81874488 is partially filled, place stop-limit for Lukoil in the amount of the filled part of the primary order, with the stop price of 271 and the order price of 270.	ACTION=NEW_STOP_ORDER; TRANS_ID=12; STOP_ORDER_KIND=ACTIVATED_BY_ORDER_SIMPLE_STOP_ORDER; BASE_ORDER_KEY=81874488; USE_BASE_ORDER_BALANCE=yes; ACTIVATE_IF_BASE_ORDER_PARTLY_FILLED=yes; PRICE=270; STOPPRICE=271; CLASSCODE=TQBR; SECCODE=LKOH; ACCOUNT=L01-00000F00; OPERATION=B; CLIENT_CODE=Q5;
'If done' take-profit and stop-limit order when order 123456 is partially filled, place a take-profit order for Lukoil at the market price, activated when the price reaches 2000, with the offset of 5 pips, and stop-limit: the stop price is 1990, execution at the market price.	ACTION=NEW_STOP_ORDER; TRANS_ID=10060; CLASSCODE= TQBR; SECCODE=LKOH; ACCOUNT=L01-00000F00; CLIENT_CODE=Q7; OPERATION=B; PRICE=2010; STOPPRICE=2000; STOP_ORDER_KIND=ACTIVATED_BY_ORDER_TAKE_PROFIT_AND_STOP_LIMIT_ORDER; OFFSET=5; OFFSET_UNITS=PRICE_UNITS; SPREAD=3; SPREAD_UNITS=PRICE_UNITS; BASE_ORDER_KEY=123456; USE_BASE_ORDER_BALANCE=YES; ACTIVATE_IF_BASE_ORDER_PARTLY_FILLED=YES; MARKET_TAKE_PROFIT=YES; STOPPRICE2=1990; MARKET_STOP_LIMIT=YES
Cancelling an order the number is 503983	CLASSCODE=TQBR; SECCODE=RU0009024277; TRANS_ID=5; ACTION=KILL_ORDER; ORDER_KEY=503983;
Cancelling an OTC order the number is 503984	CLASSCODE=TQBR; TRANS_ID=6; ACTION=KILL_NEG_DEAL; ORDER_KEY=503984;
Cancelling all orders for the client with code Q6	TRANS_ID=1; CLASSCODE=TQBR; ACTION=KILL_ALL_ORDERS; CLIENT_CODE=Q6;
Cancelling all stop orders with a buy direction	TRANS_ID=2; CLASSCODE=TQBR; ACTION=KILL_ALL_STOP_ORDERS; OPERATION=B;
Cancelling all negotiated orders for the class 'OTC: grade 1 shares'	TRANS_ID=3; CLASSCODE=PSEQ; ACTION=KILL_ALL_NEG_DEALS;

Transaction	Line
Cancelling all orders on the Moscow Exchange derivatives market placed to buy contracts on the assets of Rostelecom - ao	TRANS_ID=50; ACCOUNT=SPBFUT00001; ACTION=KILL_ALL_FUTURES_ORDERS; OPERATION=B; CLASSCODE=SPBFUT; BASE_CONTRACT=RTKM;
Moving orders on the Moscow Exchange derivatives market	ACTION=MOVE_ORDERS; TRANS_ID=333; CLASSCODE=SPBFUT; SECCODE=EBM6; FIRM_ID=SPBFUT389; MODE=1; FIRST_ORDER_NUMBER=21445064; FIRST_ORDER_NEW_PRICE=10004; FIRST_ORDER_NEW_QUANTITY=4; SECOND_ORDER_NUMBER=21445065; SECOND_ORDER_NEW_PRICE=10004; SECOND_ORDER_NEW_QUANTITY=4;
A non-addressed order for buying RusHydrp, 1 lot at 15.0 roubles, T0 settlement code, with the 'NO' attribute of cancelling active nonspecific orders. Transaction is applicable for Moscow Exchange classes	ACTION=NEW_QUOTE; TRANS_ID=779; CLASSCODE=PSEQ; SECCODE=HYDR; OPERATION=B; QUANTITY=1; PRICE=15.0; SETTLE_CODE=T0; KILL_ACTIVE_ORDERS=NO;
A non-addressed order for buying MosEnergo, 1 lot at 5.0 roubles, T0 settlement code, with the L01-00000F00 client's trading account. Transaction is applicable for SBK classes	ACTION=NEW_QUOTE; TRANS_ID=29; CLASSCODE=BPSEQ; SECCODE=MSNG; OPERATION=B; QUANTITY=1; PRICE=5.0; SETTLE_CODE=T0; ACCOUNT=L01-00000F00;
Cancelling a non-addressed order numbered 15919	ACTION=KILL_QUOTE; TRANS_ID=781; CLASSCODE=PSEQ; SECCODE=HYDR; ORDER_KEY=15919;
Import of limits on futures accounts for the account of Trader 389_011, where the 'Total' limit of open positions is 20 millions of roubles	ACTION=SET_FUT_LIMIT; TRANS_ID=22; CLASSCODE=SPBFUT; ACCOUNT=389_011; VOLUMEMN=20000000,00; KGO=0,00; USE_KGO=Y; FIRM_ID=SPBFUT389; CORRECTION=N
Confirmations of trades to be executed by a report with number 179205900	ACTION=NEW_REPORT; TRANS_ID=15; CLASSCODE=RPMA; NEG_TRADE_OPERATION=B; NEG_TRADE_NUMBER=179205900;

The format of .tro file with transaction processing results

This file is a sequence of lines each of which contains information about a single transaction. Transaction parameters are specified as 'PARAMETER_NAME=parameter_value' separated by ';'.

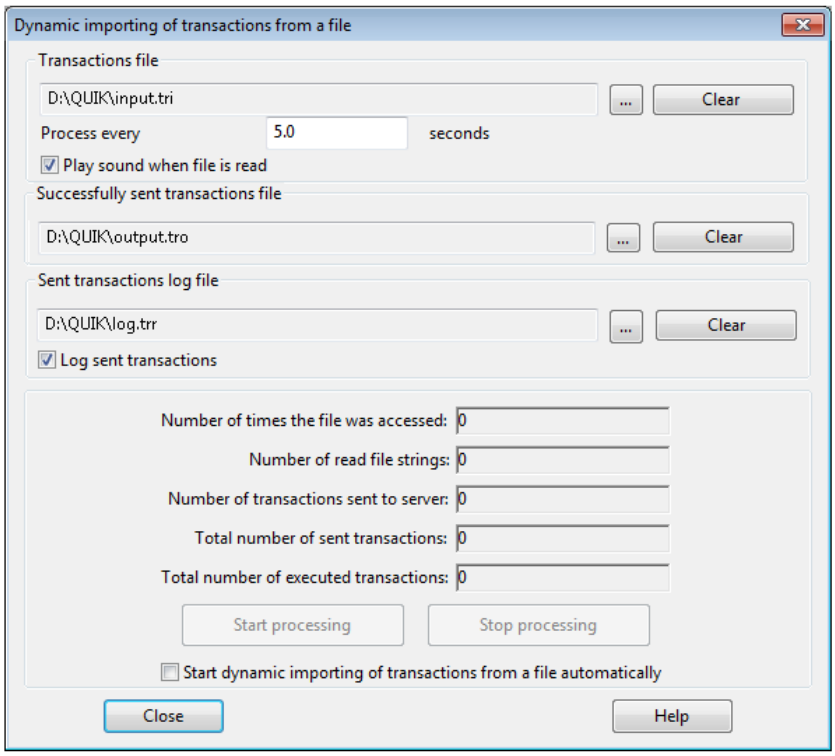
Parameters and their values:

Parameter	Description
TRANS_ID	Unique identification number of an order
STATUS	The result of the operation. Valid values: _ 0: the transaction was sent to server; _ 1: the transaction was received by the QUIK server from the client; _ 2: error during sending the transaction to the trading system due to loss of connection to MOEX gateway; the transaction cannot be sent again; _ 3: the transaction is made; _ 4: the transaction is not made in the trading system; the trading system error code will be specified in the 'DESCRIPTION' field; _ 5: the transaction was not verified by the QUIK server. For example, the user has no rights to send transactions of this type; _ 6: the transaction didn't pass the check for limits by the QUIK server; _ 10: the transaction is not supported by the trading system. For example, it was an attempt to send 'ACTION = MOVE_ORDERS' to MOEX; _ 11: the transaction didn't pass digital signature verification. For example, the server keys do not match the signature of the transaction sent to the server; _ 12: timeout for a transaction response has expired. This can happen when a transaction is sent using QPILE; _ 13: the transaction is rejected as its execution could have resulted in a cross-trade (i.e, a trade with the same client account); _ 14: the transaction failed the additional restrictions check; _ 15: the transaction is accepted after the violation of additional restrictions; _ 16: the transaction canceled by user during the additional restrictions check
TRANS_NAME	A description of the transaction, for example, 'Order entry'
DESCRIPTION	A text comment to the STATUS field containing the response from the QUIK server or trading system
ORDER_NUMBER	Registration number of an order in the trading system

An example of a line in the .tro file:

```
TRANS_ID=14;STATUS=0;TRANS_NAME="Order entry"; DESCRIPTION="Transaction sent";
TRANS_ID=14;STATUS=3;TRANS_NAME="Order entry"; DESCRIPTION="(160) Buy order N
68359610 is registered."; ORDER_NUMBER=68359610;
```

Configuration



1. Open the transaction import configuration window under **Services / Data export / import / Import transactions from file...**.
2. Enter the full path to a .tri file with transactions into the **Transactions file** field, for example, C:\quikdata\input.tri.
3. In the **Process each .. seconds** field, specify the frequency with which the file will be read. The recommended interval is 5 seconds.
4. If **Play sound when file is read** is selected, the program will beep every time the file is accessed. Use this option to monitor transactions import process.
5. Enter the full path to a .tro file into the **Successfully sent transactions file** field, for example, C:\quikdata\output.tro.
6. Enter the full path to a .trr file into the **Sent transactions log file** field, for example, C:\quikdata\log.trr.
7. If **Log sent transactions** is checked, the information about processed transactions will be written into that file.
8. The following elements (**Number of times the file was accessed**, etc) show the statistics for the import process.
9. Click **Start processing** to launch the transaction import process. Click **Stop processing** to stop this process.
10. Click **Close** to save the changes and close the configuration window. If the import process is launched, it will be performed automatically in the background.

> **Transaction data must be written into a text file as a single line. If the parameters of one transaction are written to the file in several steps, it could happen that QUIK will start reading the transaction before it has been completely written, in which case it will be rejected or processed incorrectly.**

6.10 Importing transactions via API

menu **Services / Data export/import / External transactions...**

6.10.1 Purpose

This function was designed to send transactions prepared by a client's program. This functionality is implemented in the **Trans2QUIK.dll** library. The functions from the library are described below. With the help of these functions, you can:

1. Establish or terminate a connection between the QUIK workstation and **Trans2QUIK.dll**.
2. Check the connection between the QUIK workstation and **Trans2QUIK.dll** and between the QUIK workstation and the QUIK server.
3. Send a transaction.
4. Obtain information on orders and transactions.

There are two methods of transferring transactions: synchronous and asynchronous. They are implemented in different functions.

1. During synchronous transfer, the function finishes only after a response is received from the QUIK server. Therefore, synchronous transactions can only be sent sequentially, waiting for a response to every sent transaction - this method is simpler and more suitable for programmers who has little experience in application development.
2. During asynchronous transfer, the function exits immediately. The answers about sent asynchronous transactions are received through a callback function. This function is called each time an answer about a performed or rejected transaction is received. Transaction execution statuses that are returned in the callback function are described along with the statues used for sending transactions via a file (see 6.9).

Another callback function is provided to control connections between a QUIK terminal and **Trans2QUIK.dll** and between a QUIK workstation and the QUIK server.

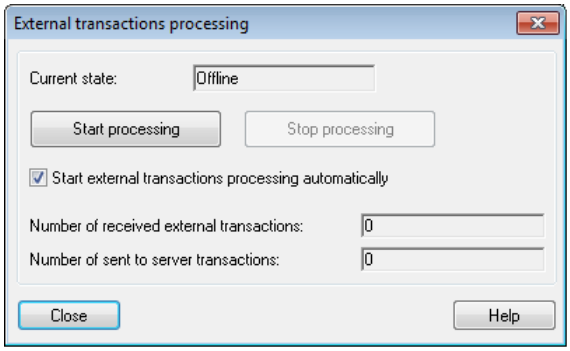
The transaction format for import via API is similar to the format used to import transaction via a file (see 6.9). The following transaction types are not supported by **Trans2quik.dll**:

- KILL_ALL_ORDERS withdraws all orders from the trading system;
- KILL_ALL_STOP_ORDERS withdraws all stop orders;
- KILL_ALL_NEG_DEALS withdraws all orders for OTC and REPO trades.

To obtain information about orders and transactions, the user must create a list of instruments to be received, separately for orders and transactions. Then the process of obtaining information with a callback is launched. When no more information about orders and transactions is received, the lists of received instruments are cleared. For further details, see 6.10.23.

6.10.2 Settings

Transaction import via API is configured in the **External transactions processing** dialogue (menu **Services / Data export/import / External transactions...**).



To start processing of external transactions, click **Start processing**. If **Start external transactions processing automatically** is checked, the service for processing external transactions starts when the QUIK workstation is run.

The statistics on sent and received transactions is displayed in the **Number of received external transactions** and **Number of transactions sent to server** fields.

6.10.3 Constants

The constants returned upon exist from functions and procedures are described below:

Constant	Description
TRANS2QUIK_SUCCESS	0
TRANS2QUIK_FAILED	1
TRANS2QUIK_QUIK_TERMINAL_NOT_FOUND	2
TRANS2QUIK_DLL_VERSION_NOT_SUPPORTED	3
TRANS2QUIK_ALREADY_CONNECTED_TO_QUIK	4
TRANS2QUIK_WRONG_SYNTAX	5
TRANS2QUIK_QUIK_NOT_CONNECTED	6
TRANS2QUIK_DLL_NOT_CONNECTED	7
TRANS2QUIK_QUIK_CONNECTED	8
TRANS2QUIK_QUIK_DISCONNECTED	9
TRANS2QUIK_DLL_CONNECTED	10
TRANS2QUIK_DLL_DISCONNECTED	11
TRANS2QUIK_MEMORY_ALLOCATION_ERROR	12
TRANS2QUIK_WRONG_CONNECTION_HANDLE	13

Constant	Description
TRANS2QUIK_WRONG_INPUT_PARAMS	14

6.10.4 Functions

The list of functions to process transactions via API:

Function	Purpose
TRANS2QUIK_CONNECT	Connects Trans2QUIK.dll to the QUIK workstation
TRANS2QUIK_DISCONNECT	Disconnects Trans2QUIK.dll from the QUIK workstation
TRANS2QUIK_IS_DLL_CONNECTED	Checks connection between Trans2QUIK.dll and the QUIK workstation
TRANS2QUIK_IS_QUIK_CONNECTED	Checks connection between the QUIK workstation and the QUIK server
TRANS2QUIK_SEND_ SYNC _TRANSACTION	Sends a synchronous transaction
TRANS2QUIK_SEND_ ASYNC _TRANSACTION	Sends an asynchronous transaction
TRANS2QUIK_CONNECTION_STATUS_CALLBACK	The prototype of the callback function used to monitor connections between Trans2QUIK.dll and a QUIK terminal in use or the QUIK workstation in use and the server
TRANS2QUIK_SET_CONNECTION_STATUS_CALLBACK	The prototype of the callback function used to handle received connection information
TRANS2QUIK_TRANSACTION_REPLY_CALLBACK	The prototype of the callback function used to handle received information about a sent transaction
TRANS2QUIK_SET_TRANSACTIONS_REPLY_CALLBACK	This function defines the TRANS2QUIK_TRANSACTION_REPLY_CALLBACK callback function
TRANS2QUIK_ORDER_STATUS_CALLBACK	The prototype of the callback function used to handle received information about an order
TRANS2QUIK_TRADE_STATUS_CALLBACK	The prototype of the callback function used to handle received information about a trade
TRANS2QUIK_SUBSCRIBE_ORDERS	Creates an instrument list by classes to receive orders
TRANS2QUIK_SUBSCRIBE_TRADES	Creates an instrument list by classes to receive trades

Function	Purpose
TRANS2QUIK_START_ORDERS	Initiates the process of receiving orders for the instruments from the list created by the TRANS2QUIK_SUBSCRIBE_ORDERS function previously
TRANS2QUIK_START_TRADES	Initiates the process of receiving trades for the instruments from the list created by the TRANS2QUIK_SUBSCRIBE_TRADES function previously
TRANS2QUIK_UNSUBSCRIBE_ORDERS	Stops the TRANS2QUIK_START_ORDERS function and clears the list of received instruments that was generated by the TRANS2QUIK_SUBSCRIBE_ORDERS function
TRANS2QUIK_UNSUBSCRIBE_TRADES	Stops the TRANS2QUIK_START_TRADES function and clears the list of received instruments that was generated by the TRANS2QUIK_SUBSCRIBE_TRADES function

6.10.5 TRANS2QUIK_CONNECT function

This function is used to connect **Trans2QUIK.dll** to the QUIK workstation.

```
long    __stdcall TRANS2QUIK_CONNECT(LPCSTR lpcstrConnectionParamsString,
long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage,
DWORD dwErrorMessageSize)
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means a successful connection; _ TRANS2QUIK_QUIK_TERMINAL_NOT_FOUND means that either INFO.EXE is not found in the specified directory or its external connections service is not running. In this case, the nTransactionExtendedErrorCode parameter will be set to 0; _ TRANS2QUIK_DLL_VERSION_NOT_SUPPORTED means that the Trans2QUIK.dll version is not supported by the specified INFO.EXE. The nTransactionExtendedErrorCode parameter will be set to 0; _ TRANS2QUIK_DLL_ALREADY_CONNECTED_TO_QUIK means that the connection is already established. The nTransactionExtendedErrorCode parameter will be set to 0; _ TRANS2QUIK_FAILED means an error during the attempt to establish connection. The nTransactionExtendedErrorCode will contain an additional error code
lpcstrConnectionParamsString	Type: pointer to String. The full path to the directory with INFO.EXE with which connection is to be established

Parameter	Description
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.6 TRANS2QUIK_DISCONNECT function

This function is used to disconnect Trans2QUIK.dll from the QUIK workstation.

long __stdcall TRANS2QUIK_DISCONNECT(long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage, DWORD dwErrorMessageSize)

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means that Trans2QUIK.dll is successfully disconnected from the QUIK workstation; _ TRANS2QUIK_FAILED means an error during the attempt to disconnect. The nTransactionExtendedErrorCode will contain an additional error code; _ TRANS2QUIK_DLL_NOT_CONNECTED means that no connection was established before the attempt to disconnect. In this case the nTransactionExtendedErrorCode variable may contain an additional error code
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.7 TRANS2QUIK_IS_QUIK_CONNECTED function

This function is used to check if connection between a QUIK terminal and the server is established.

long __stdcall TRANS2QUIK_IS_QUIK_CONNECTED (long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage, DWORD dwErrorMessageSize)

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_QUIK_CONNECTED means that a connection is established; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that a connection is not established;

Parameter	Description
	_ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and a QUIK terminal. In this case you cannot verify if the QUIK terminal is connected to server
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.8 TRANS2QUIK_IS_DLL_CONNECTED function

This function is used to check if connection between a QUIK terminal and Trans2QUIK.dll is established.

```
long __stdcall TRANS2QUIK_IS_DLL_CONNECTED
(long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage,
DWORD dwErrorMessageSize)
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_DLL_CONNECTED means that the connection between Trans2QUIK.dll and a QUIK terminal was established; _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and a QUIK terminal
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.9 TRANS2QUIK_SEND_SYNC_TRANSACTION function

Synchronous sending of transaction. During synchronous sending, the function returns a value after the results of transaction execution are received or after the QUIK terminal is disconnected from the server.

```
long __stdcall TRANS2QUIK_SEND_SYNC_TRANSACTION (LPSTR lpstTransactionString,
long* pnReplyCode, PDWORD pdwTransId, double* pdOrderNum,
LPSTR lpstrResultMessage, DWORD dwResultMessageSize,
long* nTransactionExtendedErrorCode, LPSTR lpstErrorMessage,
DWORD dwErrorMessageSize)
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means that the transaction is successfully sent to the server; _ TRANS2QUIK_WRONG_SYNTAX means that the transaction string is incorrect; _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK terminal; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK terminal and the server; _ TRANS2QUIK_FAILED means that the function failed. The nTransactionExtendedErrorCode variable can contain an additional error code
lpstrTransactionString	Type: pointer to String. A string that represents a transaction. The sting has the same format that is used to send transactions via a file
pnReplyCode	Type: pointer to Long. Contains a transaction execution status. The statuses are the same as those used for sending orders via a file (see 6.11.4)
pdwTransId	Type: pointer to Long. Contains TransID of the transaction specified by the user
pdOrderNum	Type: pointer to Double. In case of success, contains an order number in the trading system
lpstrResultMessage	Type: pointer to String. In case of success, contains a message from the trading system
dwResultMessageSize	Type: Long. Contains the length of the string pointed to by lpstrResultMessage
nTransactionExtended ErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.10 TRANS2QUIK_SEND_ASYNC_TRANSACTION function

An asynchronous transaction transfer. During asynchronous transaction sending, the function exits immediately, while the result of the transaction processing is returned via an appropriate callback function.

```
long    __stdcall TRANS2QUIK_SEND_ASYNC_TRANSACTION
(LPSTR lpstTransactionString, long* nTransactionExtendedErrorCode,
LPSTR lpstErrorMessage, DWORD dwErrorMessageSize);
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means that the transaction is successfully sent to the server; _ TRANS2QUIK_WRONG_SYNTAX means that the transaction string is incorrect;

Parameter	Description
	_ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK terminal; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK terminal and the server; _ TRANS2QUIK_FAILED means that an attempt to send a transaction failed. In this case the nTransactionExtendedErrorCode variable may contain an additional error code
lpstrTransactionString	Type: pointer to String. A string that represents a transaction. The sting has the same format that is used to send transactions via a file
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.11 TRANS2QUIK_CONNECTION_STATUS_CALLBACK

The prototype of the callback function used to monitor connections between Trans2QUIK.dll and a QUIK terminal in use or the QUIK terminal in use and the server.

```
void __stdcall TRANS2QUIK_CONNECTION_STATUS_CALLBACK
(long nConnectionEvent, long nTransactionExtendedErrorCode, LPSTR lpstrInfoMessage)
```

Parameter	Description
nConnectionEvent	Type: Long. The function can return the following values: _ TRANS2QUIK_QUIK_CONNECTED means that the connection between a QUIK terminal and the server is established; _ TRANS2QUIK_QUIK_DISCONNECTED means that the connection between a QUIK terminal and the server is closed; _ TRANS2QUIK_DLL_CONNECTED means that the connection between the DLL and a QUIK terminal is established; _ TRANS2QUIK_DLL_DISCONNECTED means that the connection between the DLL and a QUIK terminal is closed
nTransactionExtendedErrorCode	Type: Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error

6.10.12 TRANS2QUIK_SET_CONNECTION_STATUS_CALLBACK function

The prototype of the callback function used to handle received connection information.

```
long __stdcall TRANS2QUIK_SET_CONNECTION_STATUS_CALLBACK
(TRANS2QUIK_CONNECTION_STATUS_CALLBACK pfConnectionStatusCallback,
long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage,
DWORD dwErrorMessageSize)
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means that the callback function is defined; _ TRANS2QUIK_FAILED means that the callback function was not set. In this case the nTransactionExtendedErrorCode variable might contain an additional error code
TRANS2QUIK_CONNECTION_STATUS_CALLBACK	Type: function pointer. It contains the address of the function that will process the information about the connection between Trans2QUIK.dll and a QUIK terminal or a QUIK terminal and the server
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.13 TRANS2QUIK_TRANSACTION_REPLY_CALLBACK

The prototype of the callback function used to handle received information about a sent transaction.

> **IMPORTANT! You cannot send transactions asynchronously with a callback and synchronously at the same time. This is because the callback function cannot be correctly called at the moment when the function for synchronous processing of transactions has not finished.**

```
void __stdcall TRANS2QUIK_TRANSACTION_REPLY_CALLBACK(long nTransactionResult,
long nTransactionExtendedErrorCode, long nTransactionReplyCode, DWORD dwTransId,
unsigned__int64 dOrderNum, LPSTR lpstrTransactionReplyMessage,
intptr_t transReplyDescriptor)
```

Parameter	Description
nTransactionResult	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means a transaction was sent successfully; _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK terminal; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK terminal and the server; _ TRANS2QUIK_FAILED means that an attempt to send a transaction failed. In this case the nTransactionExtendedErrorCode variable may contain an additional error code
nTransactionExtendedErrorCode	Type: Long. If the callback function fails, this variable may contain an extended error code
nTransactionReplyCode	Type: Long. The pointer that contains transaction execution status. The statuses are the same as those used for sending orders via a file

Parameter	Description
dwTransId	Type: Long. The value of TransId of the registered transaction
dOrderNum	Type: unsigned__int64. The order number assigned by the trading system when the transaction is executed
lpstrTransactionReplyMessage	Type: pointer to String. A message from the trading system or QUIK server
transReplyDescriptor	Type: intptr_t. Descriptor response to a transaction may be used for the following special functions in the body of callback: – **LPTSTR _____stdcall TRANS2QUIK_TRANSACTION_REPLY_CLASS_CODE (intptr_t transReplyDescriptor)** – returns class code the transaction is entered; – **LPTSTR _____stdcall TRANS2QUIK_TRANSACTION_REPLY_SEC_CODE (intptr_t transReplyDescriptor)** – returns code of instrument the transaction is entered; – **double _____stdcall TRANS2QUIK_TRANSACTION_REPLY_PRICE (intptr_t transReplyDescriptor)** – returns trade price; – **__int64 _____stdcall TRANS2QUIK_TRANSACTION_REPLY_QUANTITY (intptr_t transReplyDescriptor)** – returns quantity; – **__int64 _____stdcall TRANS2QUIK_TRANSACTION_REPLY_BALANCE (intptr_t transReplyDescriptor)** – returns balance; – **LPTSTR _____stdcall TRANS2QUIK_TRANSACTION_REPLY_FIRMID (intptr_t transReplyDescriptor)** – returns firm code; – **LPTSTR __stdcall TRANS2QUIK_TRANSACTION_REPLY_ACCOUNT (intptr_t transReplyDescriptor)** – returns trading account; – **LPTSTR __stdcall TRANS2QUIK_TRANSACTION_REPLY_CLIENT_CODE (intptr_t transReplyDescriptor)** – returns client code; – **LPTSTR __stdcall TRANS2QUIK_TRANSACTION_REPLY_BROKERREF (intptr_t transReplyDescriptor)** – returns comment; – **LPTSTR __stdcall TRANS2QUIK_TRANSACTION_REPLY_EXCHANGE_CODE (intptr_t transReplyDescriptor)** – returns exchange number of order – **short TRANS2QUIK_TRANSACTION_REPLY_ORDERS_COUNT(TransactionReplyDescriptor tradeDescriptor)** – returns number of orders; – **EntityNumber TRANS2QUIK_TRANSACTION_REPLY_FIRST_ORDER_NUMBER_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns initial number of an order; – **EntityNumber TRANS2QUIK_TRANSACTION_REPLY_ORDER_NUMBER_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns number of an order; – **double TRANS2QUIK_TRANSACTION_REPLY_PRICE_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns price; – **Quantity TRANS2QUIK_TRANSACTION_REPLY_QUANTITY_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns quantity;

Parameter	Description
	_ **Quantity TRANS2QUIK_TRANSACTION_REPLY_BALANCE_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns balance; _ **LPTSTR TRANS2QUIK_TRANSACTION_REPLY_FIRMID_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns firm code; _ **LPTSTR TRANS2QUIK_TRANSACTION_REPLY_ACCOUNT_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns trading account; _ **LPTSTR TRANS2QUIK_TRANSACTION_REPLY_CLIENT_CODE_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns client code; _ **LPTSTR TRANS2QUIK_TRANSACTION_REPLY_BROKERREF_BY_ID(TransactionReplyDescriptor tradeDescriptor, int orderId)** – returns comment

6.10.14 TRANS2QUIK_SET_TRANSACTIONS_REPLY_CALLBACK function

Sets the callback function for receiving information about the transactions sent asynchronously.

```
long __stdcall TRANS2QUIK_SET_TRANSACTIONS_REPLY_CALLBACK
(TRANS2QUIK_TRANSACTION_REPLY_CALLBACK pfTransactionReplyCallback,
long* nTransactionExtendedErrorCode, LPSTR lpstrErrorMessage,
DWORD dwErrorMessageSize)
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_SUCCESS means that the callback function is defined; _ TRANS2QUIK_FAILED means that the callback function was not set. In this case the nTransactionExtendedErrorCode variable might contain an additional error code
TRANS2QUIK_TRANSACTION_REPLY_CALLBACK	Type: function pointer. Points to the function that will process information about a sent transaction
nTransactionExtendedErrorCode	Type: pointer to Long. In case of an error, contains an extended error code
lpstrErrorMessage	Type: pointer to String. Can contain an error message in case of an error
dwErrorMessageSize	Type: Long. Contains the length of the string pointed to by lpstrErrorMessage

6.10.15 TRANS2QUIK_ORDER_STATUS_CALLBACK function

The callback function for receiving information about order parameters.

void _stdcall TRANS2QUIK_ORDER_STATUS_CALLBACK (long nMode, DWORD dwTransID, unsigned__int64 dNumber, LPSTR lpstrClassCode, LPSTR lpstrSecCode, double dPrice, __int64 nBalance, double dValue, long nIsSell, long nStatus, intptr_t nOrderDescriptor)

Parameter	Description
nMode	Type: Long. The attribute specifying whether initial reception of orders is being performed. The Valid values: '0' refers to a new order, '1' refers to initial reception of orders is in the process, '2' refers to receiving the final order from the initial sending
dwTransID	Type: Long. The TransID of transaction that generated the order. Equals '0' if the order was not generated by a transaction from a file, or if TransID is unknown
dNumber	Type: unsigned__int64. Order number
lpstrClassCode	Type: pointer to String. Class code
lpstrSecCode	Type: pointer to String. Instrument code
dPrice	Type: Double. Order price
nBalance	Type: __int64. Unfilled amount of the order
dValue	Type: Double. Order volume
nIsSell	Type: Long. Order direction: '0' refers to 'Buying', other values mean 'Selling'
nStatus	Type: Long. Order execution status: '1' refers to 'Active', '2' refers to 'Killed', otherwise 'Filled'
nOrderDescriptor	Type: intptr_t. An order descriptor, can be used for the following special functions in the callback function: _ **int64 __stdcall TRANS2QUIK_ORDER_QTY (intptr_t nOrderDescriptor)** returns order quantity; _ **long __stdcall TRANS2QUIK_ORDER_DATE (lintptr_t nOrderDescriptor)** returns order date; _ **long T__stdcall RANS2QUIK_ORDER_TIME (intptr_t nOrderDescriptor)** returns order time; _ **long __stdcall TRANS2QUIK_ORDER_ACTIVATION_TIME (intptr_t nOrderDescriptor)** returns order activation time; _ **long __stdcall TRANS2QUIK_ORDER_WITHDRAW_TIME (intptr_t nOrderDescriptor)** returns order cancellation time; _ **long __stdcall TRANS2QUIK_ORDER_EXPIRY (intptr_t nOrderDescriptor**) returns order expiration date; _ **double __stdcall TRANS2QUIK_ORDER_ACCRUED_INT (intptr_t nOrderDescriptor)** returns accrued coupon interest of the order; _ **double __stdcall TRANS2QUIK_ORDER_YIELD (intptr_t nOrderDescriptor)** returns order yield; _ **LPSTR __stdcall TRANS2QUIK_ORDER_USERID (intptr_t nOrderDescriptor)** returns the string identifier of the trader on whose behalf the order was placed; _ **long __stdcall TRANS2QUIK_ORDER_UID (intptr_t nOrderDescriptor)** returns the UserID specified in the order; _ **LPSTR __stdcall TRANS2QUIK_ORDER_ACCOUNT (intptr_t nOrderDescriptor)** returns the trading account specified in the order; _ **LPSTR __stdcall TRANS2QUIK_ORDER_BROKERREF (intptr_t nOrderDescriptor)** returns order comment;

Parameter	Description
	_ **LPSTR __stdcall TRANS2QUIK_ORDER_CLIENT_CODE (intptr_t nOrderDescriptor)** returns the code of the client who sent the order; _ **LPSTR __stdcall TRANS2QUIK_ORDER_FIRMID (intptr_t nOrderDescriptor)** returns the string identifier of the organization to which the user who sent the order belongs; _ **__int64 __stdcall TRANS2QUIK_ORDER_VISIBLE_QTY (intptr_t nOrderDescriptor)** returns the visible quantity for Iceberg orders; _ **long __stdcall TRANS2QUIK_ORDER_PERIOD (intptr_t nOrderDescriptor)** returns the period in which the order was placed. Valid values: _ 0 – Opening; _ 1 – Normal; _ 2 – Closing. _ **FILETIME __stdcall TRANS2QUIK_ORDER_FILETIME (intptr_t nOrderDescriptor)** returns the date and time of order entry in the format YY.MM.DD HH:MM:SS.MS; _ **FILETIME __stdcall TRANS2QUIK_ORDER_WITHDRAW_FILETIME (intptr_t nOrderDescriptor)** returns the date and time of order cancellation in the format <YY.MM.DD HH:MM:SS.MS>; _ **long __stdcall TRANS2QUIK_ORDER_DATE_TIME (intptr_t nOrderDescriptor, long nTimeType)** returns the time parameters of the order depending on the value of nTimeType. The nTimeType can have the following values: _ 0: the function returns the order's entry date in the format <YYYYMMDD>; _ 1: the function returns the order's entry time in the format <HHMMSS>; _ 2: the function returns microseconds of the order's entry time, an integer from 0 to 999999; _ 3: the function returns the order's cancellation date in the format <YYYYMMDD>; _ 4: the function returns the order's cancellation time in the format HHMMSS; _ 5: the function returns microseconds of the order's cancellation time, an integer from 0 to 999999 _ **long __stdcall TRANS2QUIK_ORDER_VALUE_ENTRY_TYPE (intptr_t nOrderDescriptor)** – returns method of setting volume of order; valid values: 0 – not defined, 1 – quantity, 2 – volume; _ **long __stdcall TRANS2QUIK_ORDER_EXTENDED_FLAGS (intptr_t nOrderDescriptor)** – returns the extended flags of the order; _ **__int64 __stdcall TRANS2QUIK_ORDER_MIN_QTY (intptr_t nOrderDescriptor)** – returns the minimum possible quantity; 0 – restriction is not established; _ **long __stdcall TRANS2QUIK_ORDER_EXEC_TYPE (intptr_t nOrderDescriptor)** – returns type of order execution; 0 – value is not selected; _ **double __stdcall TRANS2QUIK_ORDER_AWG_PRICE (intptr_t nOrderDescriptor)** – returns the everage purchase price when partially executing; _ **LPTSTR __stdcall TRANS2QUIK_ORDER_REJECT_REASON (intptr_t nOrderDescriptor)** – returns reason of order rejection by broker

6.10.16 TRANS2QUIK_TRADE_STATUS_CALLBACK function

The callback function for receiving information about a trade.

long __stdcall TRANS2QUIK_TRADE_STATUS_CALLBACK (long nMode, unsigned__int64 dNumber, unsigned__int64 dOrderNum, LPSTR lpstrClassCode, LPSTR lpstrSecCode, double dPrice, __int64 nQty, double dValue, long nIsSell, intptr_t nTradeDescriptor)

Parameter	Description
nMode	Type: Long. The attribute specifying whether initial reception of trades is being performed. The Valid values: '0' means a new trade, '1' means initial reception of trades is in the process, '2' means the final trade from initial sending is received
dNumber	Type: unsigned__int64. Trade number
dOrderNum	Type: unsigned__int64. Number of the order that generated the trade
lpstrClassCode	Type: pointer to String. Class code
lpstrSecCode	Type: pointer to String. Instrument code
dPrice	Type: Double. Trade price
nQty	Type: __int64. Trade quantity
nIsSell	Type: Long. Trade direction: '0' refers to 'Buying', other values mean 'Selling'
dValue	Type: Double. Trade volume
nTradeDescriptor	Type: intptr_t. A trade descriptor, can be used for the following special functions in the callback function: _ **long __stdcall TTRANS2QUIK_TRADE_DATE (intptr_t nTradeDescriptor)** returns the trade date; _ **long __stdcall TRANS2QUIK_TRADE_SETTLE_DATE (intptr_t nTradeDescriptor)** returns the date of trade settlements; _ **long __stdcall TRANS2QUIK_TRADE_TIME (intptr_t nTradeDescriptor)** returns trade time; _ **long __stdcall TRANS2QUIK_TRADE_IS_MARGINAL (intptr_t nTradeDescriptor)** returns the flag that indicates if the trade is marginal: '0' refers to 'non-marginal', other values mean 'marginal'. _ **LPSTR __stdcall TRANS2QUIK_TRADE_CURRENCY (intptr_t nTradeDescriptor)** returns the currency of the instrument in the trade; _ **LPSTR __stdcall TRANS2QUIK_TRADE_SETTLE_CURRENCY (intptr_t nTradeDescriptor)** returns the settlement currency of the trade; _ **LPSTR __stdcall TRANS2QUIK_TRADE_SETTLE_CODE (intptr_t nTradeDescriptor)** returns the settlement code of the trade; _ **double __stdcall TRANS2QUIK_TRADE_ACCRUED_INT (intptr_t nTradeDescriptor)** returns the accrued coupon interest of the trade; _ **double __stdcall TRANS2QUIK_TRADE_YIELD (intptr_t nTradeDescriptor)** returns trade yield; _ **LPSTR __stdcall TRANS2QUIK_TRADE_USERID (intptr_t nTradeDescriptor)** returns the string identifier of the trader on which behalf the trade was executed; _ **LPSTR __stdcall TRANS2QUIK_TRADE_ACCOUNT (intptr_t nTradeDescriptor)** returns the trading account of the trade;

Parameter	Description
	_ **LPSTR __stdcall TRANS2QUIK_TRADE_BROKERREF (intptr_t nTradeDescriptor)** returns trade comment; _ **LPSTR __stdcall TRANS2QUIK_TRADE_CLIENT_CODE (intptr_t nTradeDescriptor)** returns the client code of the trade; _ **LPSTR __stdcall TRANS2QUIK_TRADE_FIRMID (intptr_t nTradeDescriptor)** returns the string identifier of the organization to which the user who executed the trade belongs; _ **LPSTR __stdcall TRANS2QUIK_TRADE_PARTNER_FIRMID (intptr_t nTradeDescriptor)** returns the string identifier of the trade partner's organization; _ **double __stdcall TRANS2QUIK_TRADE_TS_COMMISSION (intptr_t nTradeDescriptor)** returns the total commission for this trade; _ **double __stdcall TRANS2QUIK_TRADE_CLEARING_CENTER_COMMISSION (intptr_t nTradeDescriptor)** returns the clearing center's commission for this trade; _ **double __stdcall TRANS2QUIK_TRADE_EXCHANGE_COMMISSION (intptr_t nTradeDescriptor)** returns the trade exchange commission for this trade; _ **double __stdcall TRANS2QUIK_TRADE_TRADING_SYSTEM_COMMISSION (intptr_t nTradeDescriptor)** returns the commission for technical access for this trade; _ **double __stdcall TRANS2QUIK_TRADE_PRICE2 (intptr_t nTradeDescriptor)** returns the buyback price; _ **double __stdcall TRANS2QUIK_TRADE_REPO_RATE (intptr_t nTradeDescriptor)** returns the REPO rate as a percentage; _ **double __stdcall TRANS2QUIK_TRADE_REPO_VALUE (intptr_t nTradeDescriptor)** returns the REPO sum (the sum of funds attracted / provided for a REPO trade); _ **double __stdcall TRANS2QUIK_TRADE_REPO2_VALUE (intptr_t nTradeDescriptor)** returns the REPO buyback price; _ **double __stdcall TRANS2QUIK_TRADE_ACCRUED_INT2 (intptr_t nTradeDescriptor)** returns the accrued interest for buyback; _ **long __stdcall TRANS2QUIK_TRADE_REPO_TERM (intptr_t nTradeDescriptor)** returns the REPO period in calendar days; _ **double __stdcall TRANS2QUIK_TRADE_START_DISCOUNT (intptr_t nTradeDescriptor)** returns the initial discount as a percentage; _ **double __stdcall TRANS2QUIK_TRADE_LOWER_DISCOUNT (intptr_t nTradeDescriptor)** returns the lower limit of discount as a percentage; _ **double __stdcall TRANS2QUIK_TRADE_UPPER_DISCOUNT (intptr_t nTradeDescriptor**) returns the upper limit of discount as a percentage; _ **LPSTR __stdcall TRANS2QUIK_TRADE_EXCHANGE_CODE (intptr_t nTradeDescriptor)** returns the exchange code as a string; _ **LPSTR __stdcall TRANS2QUIK_TRADE_STATION_ID (intptr_t nTradeDescriptor)** returns the string identifier of the workstation; _ **long __stdcall TRANS2QUIK_TRADE_BLOCK_SECURITIES (intptr_t nTradeDescriptor)** returns the attribute indicating whether a financial instrument is blocked on a special account during a REPO operation: '0' refers to 'not blocked', other values mean 'blocked'; _ **long __stdcall TRANS2QUIK_TRADE_PERIOD (intptr_t nTradeDescriptor)** returns the period in which the trade was performed. The Valid values: '0' refers to 'Opening', '1' refers to 'Normal', '2' refers to 'Closing'.

Parameter	Description
	_ **FILETIME __stdcall TRANS2QUIK_TRADE_FILETIME (intptr_t nTradeDescriptor)** returns the date and time of a trade in the format YY.MM.DD HH:MM:SS.MS; _ **long __stdcall TRANS2QUIK_TRADE_DATE_TIME (intptr_t nTradeDescriptor, long nTimeType)** returns the time parameters of a trade depending on the value of nTimeType. The nTimeType can have the following values: _ 0: the function returns the trade date in the format <YYYYMMDD>; _ 1: the function returns the trade time in the format <HHMMSS>; _ 2: the function returns microseconds of the trade time, an integer from 0 to 999999 _ **long __stdcall TRANS2QUIK_TRADE_KIND (intptr_t nTradeDescriptor)** returns the trade's type. Valid values: _ 1 - regular; _ 2 - negotiated; _ 3 - initial placement; _ 4 - cash / instruments transfer; _ 5 - negotiated trade of the first REPO leg; _ 6 - swap transaction settlement trade; _ 7 - OTC swap transaction settlement trade; _ 8 - dual currency basket settlement trade; _ 9 - OTC dual currency basket settlement trade. _ **double __stdcall TRANS2QUIK_TRADE_BROKER_COMMISSION (intptr_t nTradeDescriptor)** – returns amount of broker commission; _ **long __stdcall TRANS2QUIK_TRADE_TRANSID (intptr_t nTradeDescriptor)** – returns TRANS_ID of order generate a trade

6.10.17 TRANS2QUIK_SUBSCRIBE_ORDERS function

This function creates a list of classes and instruments that is used to receive orders for these instruments.

long __stdcall TRANS2QUIK_SUBSCRIBE_ORDERS (LPSTR lpstrClassCode, LPSTR lpstrSeccodes)

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK workstation. In this case, you cannot subscribe to orders; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK workstation and the server. In this case, you cannot subscribe to orders; _ TRANS2QUIK_SUCCESS means a successful subscription; _ TRANS2QUIK_FAILED means an attempt to subscribe failed
lpstrClassCode	Type: pointer to String. The code of the class to request orders. If both input parameters are empty strings, orders for all available instruments will be requested
lpstrSeccodes	Type: pointer to String. The list of instrument codes for which orders will be requested, separated by '|'. If an empty string is specified, orders will be requested for the class specified in the lpstrClassCode parameter

6.10.18 TRANS2QUIK_SUBSCRIBE_TRADES function

This function creates a list of classes and instruments that is used to receive trades for these instruments.

long __stdcall TRANS2QUIK_SUBSCRIBE_TRADES (LPSTR lpstrClassCode, LPSTR lpstrSeccodes)

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK workstation. In this case, you cannot subscribe to orders; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK workstation and the server. In this case, you cannot subscribe to orders; _ TRANS2QUIK_SUCCESS means a successful subscription; _ TRANS2QUIK_FAILED means an attempt to subscribe failed
lpstrClassCode	Type: pointer to String. The code of the class to request trades. If both input parameters are empty strings, trades for all available instruments will be requested
lpstrSeccodes	Type: pointer to String. The list of instrument codes for which trades will be requested, separated by '|'. If an empty string is specified, trades will be requested for the class specified in the lpstrClassCode parameter

6.10.19 TRANS2QUIK_START_ORDERS function

This function launches the process of receiving orders for classes and instruments defined by the **TRANS2QUIK_SUBSCRIBE_ORDERS** function.

void __stdcall TRANS2QUIK_START_ORDERS (TRANS2QUIK_ORDER_STATUS_CALLBACK pfnOrderStatusCallback)

Parameter	Description
TRANS2QUIK_ORDER_STATUS_CALLBACK	A pointer to a user callback function used to obtain information about orders

6.10.20 TRANS2QUIK_START_TRADES function

This function launches the process of receiving trades and parameters defined by the **TRANS2QUIK_SUBSCRIBE_TRADES** function.

void __stdcall TRANS2QUIK_START_TRADES (TRANS2QUIK_TRADE_STATUS_CALLBACK pfnTradesStatusCallback)

Parameter	Description
TRANS2QUIK_TRADE_STATUS_CALLBACK	A pointer to a user callback function used to obtain information about trades

6.10.21 TRANS2QUIK_UNSUBSCRIBE_ORDERS function

This function stops the **TRANS2QUIK_START_ORDERS** function and clears the list of received instruments that was created by the **TRANS2QUIK_SUBSCRIBE_ORDERS** function.

```
long __stdcall TRANS2QUIK_UNSUBSCRIBE_ORDERS ()
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK workstation. The operation cannot be performed; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK workstation and the server. The operation cannot be performed; _ TRANS2QUIK_SUCCESS means the operation was performed successfully; _ TRANS2QUIK_FAILED means an attempt to perform the operation failed

6.10.22 TRANS2QUIK_UNSUBSCRIBE_TRADES function

This function stops the **TRANS2QUIK_START_TRADES** function and clears the list of received instruments that was generated by the **TRANS2QUIK_SUBSCRIBE_TRADES** function.

```
long __stdcall TRANS2QUIK_UNSUBSCRIBE_TRADES ()
```

Parameter	Description
Result	Type: Long. The function can return the following values: _ TRANS2QUIK_DLL_NOT_CONNECTED means that there is no connection between Trans2QUIK.dll and the QUIK workstation. The operation cannot be performed; _ TRANS2QUIK_QUIK_NOT_CONNECTED means that there is no connection between the QUIK workstation and the server. The operation cannot be performed; _ TRANS2QUIK_SUCCESS means the operation was performed successfully; _ TRANS2QUIK_FAILED means an attempt to perform the operation failed

6.10.23 Obtaining information about orders and trades

To obtain information about orders and transactions, the user must create a list of instruments to be received. To do this, call appropriate subscription functions:

TRANS2QUIK_SUBSCRIBE_ORDERS for orders and **TRANS2QUIK_SUBSCRIBE_TRADES** for trades.

The list of instruments for both orders and trades can be generated using two methods: subscribing on the entire list for which the user has rights, or listing classes one by one. For example, if you need information for only two classes, TQNL and TQBR, and you are only interested in orders for the LKOH instrument in the class TQBR, the subscription should look as follows:

```
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQNL", "");
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "LKOH");
```

To list several instruments in a class, use '|'. For example:

```
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "LKOH|AFLT");
```

The '|' separator cannot be used for listing classes of instruments.

If the subscription function is called with a number of instruments that are already present in the list, such call will be ignored, for example:

```
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "");
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "LKOH|AFLT");
```

First, a subscription to orders for all available classes and instruments was created, then, an attempt to subscribe to information for one particular class. This call will be ignored, as it does not add new instruments to the list of orders to be received. The following case will produce the same result:

```
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "");
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "LKOH|AFLT");
```

To launch the process of receiving information on trades and orders, the user must call the **TRANS2QUIK_START_ORDERS** function for orders and **TRANS2QUIK_START_TRADES** for trades.

After calling these functions, your QUIK terminal will start transferring to **Trans2QUIK.dll** the information about orders and trades with the specified callback functions. Information about already received trades and orders will be transferred first (the **nMode** parameter in the callback functions will be different from zero), and information about new orders and trades will be transferred as they arrive (the **nMode** parameter in callback functions will be zero).

To stop the process of receiving information on trades and orders, the user must call the corresponding unsubscribe functions: **TRANS2QUIK_UNSUBSCRIBE_ORDERS** for orders and **TRANS2QUIK_UNSUBSCRIBE_TRADES** for trades. These functions have no parameters and, when called, stop the process of receiving of information for all trades and orders. These functions as well as the functions for subscribing to trades and orders can be called multiple times. Example:

```
TRANS2QUIK_SUBSCRIBE_ORDERS ("", "");
TRANS2QUIK_START_ORDERS ();
TRANS2QUIK_SUBSCRIBE_TRADES ("", "");
TRANS2QUIK_START_TRADES ();
…
TRANS2QUIK_UNSUBSCRIBE_ORDERS ();
TRANS2QUIK_UNSUBSCRIBE_TRADES ();
TRANS2QUIK_SUBSCRIBE_ORDERS ("TQBR", "LKOH");
TRANS2QUIK_START_ORDERS ();
TRANS2QUIK_SUBSCRIBE_TRADES ("TQBR", "LKOH");
TRANS2QUIK_START_TRADES ();
```

6.11 APPENDIX

6.11.1 Data formats for configuring export via ODBC

Parameter	Format	Parameter	Format	Parameter	Format
Quotes Table					
Instrument[class]*	VARCHAR(200)	Close price	DECIMAL(15,6)	Strike	DECIMAL(15,6)
Instrument	VARCHAR(150)	Prev. VWAP	DECIMAL(15,6)	Price step	DECIMAL(15,6)
Instrument (s.n.)	VARCHAR(20)	Auction	VARCHAR(32)	Settle price	DECIMAL(15,6)
Instrument code	VARCHAR(12)	Accrued int.	DECIMAL(15,8)	Last trade qty	INTEGER
ISIN code	VARCHAR(15)	Yield	DECIMAL(15,6)	Change to prev. session	DECIMAL(15,6)
Reg. number	VARCHAR(30)	Coupon payment	DECIMAL(15,6)	prevsettleprice	DECIMAL(15,6)
Class	VARCHAR(128)	Yld. prev. VWAP	DECIMAL(15,6)	Price var. limit	DECIMAL(15,6)
Class code	VARCHAR(12)	Yld. VWAP	DECIMAL(15,6)	Price var. limit T+1	DECIMAL(15,6)
Trade date	VARCHAR(15)	Price-VWAP	DECIMAL(15,6)	Active order limit	DECIMAL(15,6)
Expiration	VARCHAR(15)	Close period	DECIMAL(15,6)	Neg. deal value.	DECIMAL(15,0)
Time to maturity	INTEGER	Close yield	DECIMAL(15,6)	Neg. deal qty	INTEGER
Face-value	DECIMAL(15,6)	% of last change	DECIMAL(15,6)	Current value	DECIMAL(15,6)
Currency	VARCHAR(4)	Pr. mark. price	DECIMAL(15,6)	Close value	DECIMAL(15,6)
Scale	INTEGER	Market price	DECIMAL(15,6)	Prev. close time	VARCHAR(15)
Price step	DECIMAL(15,6)	Market price 2	DECIMAL(15,6)	Open value	DECIMAL(15,6)
Status	VARCHAR(32)	Adm.quotation	DECIMAL(15,6)	Cur-Open index	DECIMAL(15,6)
Lot size	INTEGER	Next coup. pay	VARCHAR(15)	Cur-Close index	DECIMAL(15,6)
Session status	VARCHAR(32)	Coupon period	INTEGER	Last bid	DECIMAL(19,0)
Type	VARCHAR(12)	Price of offer	DECIMAL(15,6)	Last offer	DECIMAL(19,0)

Parameter	Format	Parameter	Format	Parameter	Format
Bid	DECIMAL(15,6)	Offer date	VARCHAR(15)	Prev.close price	DECIMAL(19,6)
Bid vol.	INTEGER	Issue size	INTEGER	Agg. rate	DECIMAL(15,6)
Total bid vol.	INTEGER	Prev. trd date	VARCHAR(15)	Type of price	VARCHAR(16)
Num. bids	INTEGER	Duration	DECIMAL(15,6)	Clear. quote	DECIMAL(15,6)
Offer	DECIMAL(15,6)	Off. op. price	DECIMAL(15,6)	Auct. beg time	VARCHAR(15)
Off. vol.	INTEGER	Off. cur. price	DECIMAL(15,6)	Auct. end time	VARCHAR(15)
Tot. off. vol.	INTEGER	Off. close price	DECIMAL(15,6)	Start ev.sess.	VARCHAR(15)
Num. offers	INTEGER	Price type	VARCHAR(128)	End ev.sess.	VARCHAR(15)
Open	DECIMAL(15,6)	Trend	DECIMAL(15,6)	Start morn. sess.	VARCHAR(15)
High	DECIMAL(15,6)	Prev. adm. quote	DECIMAL(19,6)	End morn. sess.	VARCHAR(15)
Low	DECIMAL(15,6)	Open pos.	INTEGER	curstepprice	VARCHAR(16)
Last	DECIMAL(15,6)	Highest price	DECIMAL(15,6)	realvmprice	DECIMAL(15,6)
Change	DECIMAL(15,6)	Lowest price	DECIMAL(15,6)	Marg	VARCHAR(16)
Quantity	INTEGER	Init. buy margin	DECIMAL(15,6)	expdate	VARCHAR(15)
Time	VARCHAR(15)	Init. sell margin	DECIMAL(15,6)	steppricecl	DECIMAL(15,6)
Vol.today	INTEGER	Change time	VARCHAR(15)	Min. cut. pr. time	VARCHAR(15)
Val.today	DECIMAL(15,0)	Covered margin	DECIMAL(15,6)	Prev.lot size	INTEGER
Value	DECIMAL(15,6)	Not cov. margin	DECIMAL(15,6)	D.lot.size ch.	VARCHAR(15)
VWAP	DECIMAL(15,6)	Opt. kind	VARCHAR(16)	Offer date	VARCHAR(15)
High. bid	DECIMAL(15,6)	Base asset	VARCHAR(13)	Basic rate	DECIMAL(19,4)
Low. offer	DECIMAL(15,6)	Volatility	DECIMAL(15,6)	CFI code	VARCHAR(12)
Num. trades	INTEGER	Theor.price	DECIMAL(15,6)	Availability of any orders on the dark pool auction	VARCHAR(8)
Client portfolio table					
Firm	VARCHAR(12)	Current Leverage	DECIMAL(20,6)	LimBuy	DECIMAL(20,6)
Client code	VARCHAR(12)	Margin	DECIMAL(20,6)	LimSell	DECIMAL(20,6)

Parameter	Format	Parameter	Format	Parameter	Format
HighRisk	VARCHAR(12)	LimAll	DECIMAL(20,6)	LimNonMargin	DECIMAL(20,6)
Client type	VARCHAR(3)	AvLimAll	DECIMAL(20,6)	LimBuyAsset	DECIMAL(205, 6)
Fut. trade account	VARCHAR(12)	LockedBuy	DECIMAL(20,6)	Pos. margin	DECIMAL(20,6)
InAssets	DECIMAL(20,6)	LockedBuyMargin	DECIMAL(20,6)	Orders margin	DECIMAL(20,6)
Leverage	DECIMAL(20,6)	LockedBuyAsset	DECIMAL(20,6)	Variat. margin	DECIMAL(20,6)
Open limit	DECIMAL(20,6)	LockedSell	DECIMAL(20,6)	Assets/Curr.net pos.	DECIMAL(20,6)
ValShort	DECIMAL(20,6)	LockedBuyNonMargin	DECIMAL(20,6)	Total money balance	DECIMAL(20,6)
ValLong	DECIMAL(20,6)	OpenAllAssets	DECIMAL(20,6)	Total locked money	DECIMAL(20,6)
ValLongMargin	DECIMAL(20,6)	AllAssets	DECIMAL(20,6)	Calc. params	VARCHAR(10)
ValLongAsset	DECIMAL(20,6)	ProfitLoss	DECIMAL(20,6)	Short (net)	DECIMAL(20,6)
Portfolio value	DECIMAL(20,6)	RateChange	DECIMAL(15,8)	Long (net)	DECIMAL(20,6)
Position on date	VARCHAR(5)	Min.margin	DECIMAL(20,6)	Init.margin	DECIMAL(20,6)
Corr.margin	DECIMAL(20,6)	Status	VARCHAR(12)	Demand	DECIMAL(20,6)
Funds level	DECIMAL(20,6)	Haircuts	DECIMAL(20,6)	Assets w / o HC	DECIMAL(20,6)
Status coef.	DECIMAL(20,6)	OpenPosLimBegin	DECIMAL(20,6)	OpenPosLim	DECIMAL(20,6)
PlanNetPos	DECIMAL(20,6)	CurrNetPos	DECIMAL(20,6)	AccVarMarg	DECIMAL(20,6)
AccVarMargIntCl	DECIMAL(20,6)	AccruedInt	DECIMAL(20,6)	OptLiquidVal	DECIMAL(20,6)
FutMrkAssets	DECIMAL(20,6)	TotalMrkAssets	DECIMAL(20,6)	CurrDebdtFut	DECIMAL(20,6)
FundsAdeq	DECIMAL(20,6)	FundsAdeq (OpenPos)	DECIMAL(20,6)	ColLiquidCoef	DECIMAL(20,6)
ExColLiquidCoef	DECIMAL(20,6)	Cash Leverage	DECIMAL(20,6)	PosTypeFutMrk	VARCHAR(12)

Parameter	Format	Parameter	Format	Parameter	Format
Time and Sales Table					
Number	DECIMAL(19,0)	Instrument code	VARCHAR(12)	Yield	DECIMAL(15,6)
Trading date	VARCHAR(20)	Class	VARCHAR(130)	Accrued profit	DECIMAL(15,2)
Date	VARCHAR(20)	Class code	VARCHAR(12)	REPO rate (%)	DECIMAL(15,6)
Time	VARCHAR(15)	Price	DECIMAL(15,6)	REPO sum	DECIMAL(15,2)
Time(microsec)	INTEGER	Qty	INTEGER	REPO buyback value	DECIMAL(15,8)
Period	VARCHAR(20)	Volume	DECIMAL(19,8)	REPO period	INTEGER
Instrument (s.n.)	VARCHAR(20)	Operation	VARCHAR(12)	Open interest	DECIMAL(15,0)
Instrument	VARCHAR(150)	Settlement code	VARCHAR(5)	Exchange code	VARCHAR(64)
Orders table					
Number	DECIMAL(19,0)	Depo account	VARCHAR(12)	Status	VARCHAR(10)
Exchange code	VARCHAR(64)	Price	DECIMAL(15,6)	Trans ID	INTEGER
Trading date	VARCHAR(20)	Qty	INTEGER	Settlement code	VARCHAR(12)
Date	VARCHAR(20)	Visible qty	DECIMAL(19,0)	Buyback price	DECIMAL(15,6)
Sent(time)	VARCHAR(10)	Balance	INTEGER	Market-maker's order	VARCHAR(3)
Sent(microsec)	INTEGER	Volume	DECIMAL(15,8)	BankAccID	VARCHAR(12)
Period	VARCHAR(20)	Currency	VARCHAR(4)	Value entry type	VARCHAR(16)
Activation time	VARCHAR(10)	Yield	DECIMAL(15,6)	REPO period	INTEGER
Killed date	VARCHAR(20)	Accrued interest	DECIMAL(15,2)	REPO sum	DECIMAL(15,8)
Killed(time)	VARCHAR(10)	Trader	VARCHAR(12)	REPO buyback value	DECIMAL(15,8)
Killed(microsec)	INTEGER	Dealer	VARCHAR(12)	REPO sum balance	DECIMAL(15,8)
Instrument (s.n.)	VARCHAR(20)	UID	INTEGER	Start discount (%)	DECIMAL(15,6)

Parameter	Format	Parameter	Format	Parameter	Format
Instrument	VARCHAR(150)	Client code	VARCHAR(20)	Reject reason	VARCHAR(128)
Instrument code	VARCHAR(12)	Comment	VARCHAR(20)	Execution type	VARCHAR(64)
Class	VARCHAR(130)	Linked order	DECIMAL(19,0)	Min qty	INTEGER
Class code	VARCHAR(12)	Expiration	VARCHAR(15)	Stop order	DECIMAL(19,0)
Side	VARCHAR(10)	Type	VARCHAR(10)	Expire time	VARCHAR(8)
W.avg.price	DECIMAL(15,6)	UID canceled order	DECIMAL(15,0)	Extended status	VARCHAR(16)
Base currency quantity	DECIMAL(15,6)	Quote currency quantity	DECIMAL(15,6)		
Base currency	VARCHAR(16)	Quote currency	VARCHAR(16)		
Stop orders table					
Number	DECIMAL(19,0)	Stop price direction	VARCHAR(3)	Active from	VARCHAR(12)
Date	VARCHAR(10)	Stop price	DECIMAL(15,6)	Active to	VARCHAR(12)
Time	VARCHAR(12)	Stop limit price direction	VARCHAR(3)	Type	VARCHAR(4)
Cancellation time	VARCHAR(10)	Stop-limit price	DECIMAL(15,6)	Status	VARCHAR(12)
Stop order type description	VARCHAR(128)	Price	DECIMAL(15,6)	Result	VARCHAR(64)
Description of the stop order type	VARCHAR(128)	Market stop-limit	VARCHAR(2)	Linked order	DECIMAL(19,0)
Instrument (s.n.)	VARCHAR(16)	Qty	DECIMAL(15,0)	Linked order price	DECIMAL(15,6)
Instrument	VARCHAR(128)	Act. qty	DECIMAL(15,0)	Trans ID	DECIMAL(15,0)
Instrument code	VARCHAR(12)	Filled qty.	DECIMAL(15,0)	Offset from min / max	DECIMAL(15,6)
Class	VARCHAR(128)	Dealer	VARCHAR(12)	Offset units	VARCHAR(1)
Class code	VARCHAR(12)	UID	DECIMAL(15,0)	Protective spread	DECIMAL(15,6)

Parameter	Format	Parameter	Format	Parameter	Format
Operation	VARCHAR(8)	Client code	VARCHAR(128)	Spread units	VARCHAR(1)
Depo account	VARCHAR(12)	Comment	VARCHAR(20)	Take-profit at market price	VARCHAR(3)
Stop price instr.	VARCHAR(128)	Order number	DECIMAL(19,0)	Primary order	DECIMAL(19,0)
Stop-price instr. code	VARCHAR(12)	Condition trade	DECIMAL(15,0)	Server	VARCHAR(12)
Stop price instr. class	VARCHAR(128)	Expiration	VARCHAR(10)	Cancellation date	VARCHAR(10)
Stop price instr. class code	VARCHAR(12)	Active in time	VARCHAR(3)	UID canceled order	DECIMAL(15,0)
Trades table					
Number	DECIMAL(19,0)	Value	DECIMAL(15,8)	TS Commission	DECIMAL(15,8)
Exchange code	VARCHAR(68)	Currency	VARCHAR(4)	Clearing centre commission	DECIMAL(15,6)
Trading date	VARCHAR(20)	Settlement currency	VARCHAR(4)	Exchange commission	DECIMAL(15,6)
Settlement date	VARCHAR(10)	Settle code	VARCHAR(5)	TC commission	DECIMAL(15,6)
Time	VARCHAR(15)	Yield	DECIMAL(15,6)	Profit (%) for buyback date	DECIMAL(15,6)
Order number	DECIMAL(19,0)	Accrued profit	DECIMAL(15,8)	REPO sum	DECIMAL(15,8)
Instrument (s.n.)	VARCHAR(20)	Trader	VARCHAR(12)	REPO buyback value	DECIMAL(15,8)
Instrument	VARCHAR(150)	Station ID	VARCHAR(36)	REPO period	INTEGER
Instrument code	VARCHAR(12)	Dealer	VARCHAR(12)	Start discount (%)	DECIMAL(15,6)
Class	VARCHAR(130)	Trader's org.	VARCHAR(128)	Lower discount (%)	DECIMAL(15,6)
Class code	VARCHAR(12)	Client code	VARCHAR(20)	Upper discount (%)	DECIMAL(15,6)
Trade type	VARCHAR(12)	Broker reference	VARCHAR(20)	Block instruments	VARCHAR(3)
Trade side	VARCHAR(10)	Partner	VARCHAR(12)	Trade date	VARCHAR(20)

Parameter	Format	Parameter	Format	Parameter	Format
Trade account	VARCHAR(12)	Partner's org.	VARCHAR(129)	Kind of trade	VARCHAR(64)
Price	DECIMAL(15,6)	Period	VARCHAR(20)	BankAccID	VARCHAR(12)
Time (μs)	INTEGER	Buyback price	DECIMAL(15,6)	Linked trade	DECIMAL(19,0)
Quantity	INTEGER	REPO rate (%)	DECIMAL(15,6)	Iceberg order	VARCHAR(4)
Settlement currency	VARCHAR(4)	Transaction ID	DECIMAL(15,0)	Canceled (microsec)	VARCHAR(8)
Broker commission	DECIMAL(15,6)	Canceled UID	DECIMAL(15,0)	System reference	VARCHAR(15)
Clearing firm	VARCHAR(18)	Cancellation date	VARCHAR(10)	Status	VARCHAR(15)
Clearing bank account	VARCHAR(18)	Canceled (time)	VARCHAR(8)	Prefferable instrument	VARCHAR(15)
Base currency quantity	DECIMAL(15,6)	Quote currency quantity	DECIMAL(15,6)		
Base currency	VARCHAR(16)	Quote currency	VARCHAR(16)		
Table of cash positions					
Firm	VARCHAR(12)	Incoming position	DECIMAL(15,6)	Total	DECIMAL(15,6)
Currency	VARCHAR(5)	Incoming limit	DECIMAL(15,6)	Available	DECIMAL(15,6)
Position code	VARCHAR(5)	Current balance	DECIMAL(15,6)	Balance	DECIMAL(15,6)
Client code	VARCHAR(12)	Current limit	DECIMAL(15,6)		
Position on date	VARCHAR(5)	Reserved	DECIMAL(15,6)		
Table of positions in instruments					
Firm	VARCHAR(12)	Incoming position	INTEGER	Available	INTEGER
Instrument	VARCHAR(150)	Incoming limit	INTEGER	Balance	INTEGER
Instrument code	VARCHAR(12)	Current balance	INTEGER	WA.position price	DECIMAL(15,6)
Depo account	VARCHAR(12)	Current limit	INTEGER	Reserved buy	DECIMAL(15,6)
Client code	VARCHAR(12)	Reserved	INTEGER		

Parameter	Format	Parameter	Format	Parameter	Format
Position on date	VARCHAR(5)	Total	INTEGER		
Table of client account positions (futures)					
Firm	VARCHAR(12)	Open short. pos	INTEGER	Cur.net pos. appr.	DECIMAL(15,6)
Trade account	VARCHAR(12)	Cur. long pos.	INTEGER	Plan. net pos.	DECIMAL(15,6)
Instrument code	VARCHAR(12)	Cur. short. pos.	INTEGER	Variat. margin	DECIMAL(15,6)
Short name	VARCHAR(150)	Cur.net pos.	INTEGER	Effect. pos. price	DECIMAL(15,6)
Type	VARCHAR(35)	Active on buy	INTEGER	Total v. margin	DECIMAL(15,2)
Open long pos.	INTEGER	Position value	DECIMAL(19,4)	Real v. margin	DECIMAL(15,2)
Expiration date	VARCHAR(20)	Active on sell	INTEGER		
Table of client account limits (futures)					
Firm	VARCHAR(12)	Open limit	DECIMAL(15,6)	Cur. net positions (for orders)	DECIMAL(19,4)
Trade account	VARCHAR(12)	Cur. net pos.	DECIMAL(15,6)	Cur. net positions (for open positions)	DECIMAL(19,4)
Limit type	VARCHAR(128)	Plan. net pos.	DECIMAL(15,6)	Coeff. of client marginal requirements	DECIMAL(19,6)
Liquid. coef.	DECIMAL(15,6)	Stock exchange tax	DECIMAL(19,4)	Holding currency	VARCHAR(4)
Options premium	DECIMAL(19,4)	Variation margin for positions	DECIMAL(15,6)	Real v. margin	DECIMAL(15,2)
Prev. open limit	DECIMAL(15,6)	Accrued profit	DECIMAL(15,6)		
Level II Quotes Table					
Buy yield	DECIMAL(15,6)	Price	DECIMAL(15,6)	Sell total volume	INTEGER

Parameter	Format	Parameter	Format	Parameter	Format
Buy total volume	INTEGER			Sell yield	DECIMAL(15,6)
Own buy	INTEGER	Sell	INTEGER		
Buy	INTEGER	Own sell	INTEGER		
Table of order reports for NDM trades					
Number	DECIMAL(19,0)	Trader	VARCHAR(12)	Qty	INTEGER
Date	VARCHAR(10)	Dealer	VARCHAR(12)	Value	DECIMAL(15,8)
Sent(time)	VARCHAR(8)	Trader's org.	VARCHAR(128)	Commission	DECIMAL(15,6)
Killed(time)	VARCHAR(8)	Depo account	VARCHAR(12)	Side	VARCHAR(32)
Class	VARCHAR(128)	Partner	VARCHAR(12)	Status	VARCHAR(32)
Instrument (s.n.)	VARCHAR(16)	Partner's org.	VARCHAR(128)	Report type	VARCHAR(32)
Instrument	VARCHAR(128)	Partner's depo account	VARCHAR(12)	Report kind	VARCHAR(32)
Compensation value	DECIMAL(15,6)	Compensation direction	VARCHAR(12)		
Table of trades for execution					
Number	DECIMAL(19,0)	Status	VARCHAR(32)	Upper discount (%)	DECIMAL(15,6)
Order number	DECIMAL(19,0)	Accrued profit	DECIMAL(15,8)	Block instruments	VARCHAR(3)
Date	VARCHAR(10)	Price first REPO part	DECIMAL(15,6)	Execute	VARCHAR(3)
Settlement date	VARCHAR(10)	Buyback price	DECIMAL(15,6)	Execute tomorrow	VARCHAR(3)
Class	VARCHAR(128)	Trade number 1st REPO part	DECIMAL(19,0)	Type	VARCHAR(50)
Instrument ticker	VARCHAR(16)	REPO rate (%)	DECIMAL(15,6)	Direction	VARCHAR(12)
Instrument	VARCHAR(128)	Settle code	VARCHAR(12)	Discount after payment (%)	DECIMAL(15,6)
Side	VARCHAR(12)	Report	DECIMAL(15,0)	Quantity after payment	INTEGER

Parameter	Format	Parameter	Format	Parameter	Format
Client code	VARCHAR(128)	Partner's report	DECIMAL(15,0)	REPO sun after payment	DECIMAL(15,8)
Comment	VARCHAR(128)	TS Commission	DECIMAL(15,6)	REPO buyback sum after payment	DECIMAL(15,8)
Dealer	VARCHAR(12)	Balance	INTEGER	REPO return sum after payment	DECIMAL(15,8)
Trader's org.	VARCHAR(128)	Execution time	VARCHAR(10)	Date of settlement	VARCHAR(10)
Depo account	VARCHAR(12)	Sum of liabilities	DECIMAL(15,8)	Clearing status	VARCHAR(32)
Partner	VARCHAR(12)	REPO sum	DECIMAL(15,8)	Type of clearing	VARCHAR(32)
Partner's org.	VARCHAR(128)	REPO period	INTEGER	Report comission	DECIMAL(15,6)
Partner's depo account	VARCHAR(12)	REPO buyback value	DECIMAL(15,8)	Coupon payment	DECIMAL(15,6)
Price	DECIMAL(15,6)	REPO return value	DECIMAL(15,8)	Date of coupon payment	VARCHAR(10)
Qty	INTEGER	Discount (%)	DECIMAL(15,6)	Principal debt payment	DECIMAL(15,6)
Value	DECIMAL(15,8)	Lower discount (%)	DECIMAL(15,6)	Confirmed	VARCHAR(12)
Date of principal debt payment	VARCHAR(10)	Settle currency	VARCHAR(5)	Confirmed by counterparty	VARCHAR(12)
Number of instruction	DECIMAL(15,0)	Confirmation time	VARCHAR(15)	Position code	VARCHAR(12)
Qty for execution		Compensation value	DECIMAL(15,6)		
Bank	VARCHAR(12)	Parent trade number	DECIMAL(19,0)		
Table liabilities and claims on assets					
Firm	VARCHAR(12)	Class code	VARCHAR(12)	Sell qty	DECIMAL(15,0)
Instrument code	VARCHAR(12)	Class	VARCHAR(128)	Netto	DECIMAL(15,2)

Parameter	Format	Parameter	Format	Parameter	Format
Trading account	VARCHAR(12)	Depo account	VARCHAR(12)	Debit	DECIMAL(15,2)
Settle date	VARCHAR(10)	BankAccId	VARCHAR(12)	Credit	DECIMAL(15,2)
Instrument (s.n.)	VARCHAR(16)	Quantity	DECIMAL(15,0)		
Instrument	VARCHAR(128)	Buy qty	DECIMAL(15,0)		
Table of cash liabilities and claims					
Firm	VARCHAR(12)	Settle date	VARCHAR(10)	Debit	DECIMAL(15,2)
BankAccId	VARCHAR(12)	Netto	DECIMAL(15,2)	Credit	DECIMAL(15,2)
Interest rate risk parameters table					
Instrument code	VARCHAR(12)	Range start	DECIMAL(15,0)	REPO settlement rate, %	DECIMAL(15,2)
Instrument (s.n.)	VARCHAR(16)	Range end	DECIMAL(15,0)	High rate, %	DECIMAL(15,2)
Instrument	VARCHAR(128)	Discount, %	DECIMAL(15,1)	Range start, rur	DECIMAL(15,2)
Class code	VARCHAR(12)	Low rate, rur	DECIMAL(15,6)	Range end, rur	DECIMAL(15,2)
Class	VARCHAR(128)	REPO settlement rate, rur	DECIMAL(15,6)	Range	INTEGER
Settle date	VARCHAR(10)	High rate, rur	DECIMAL(15,6)	Low rate, %	DECIMAL(15,2)
Market risk parameters table					
Instrument code	VARCHAR(12)	Range	INTEGER	High rate, rur	DECIMAL(15,6)
Instrument (s.n.)	VARCHAR(16)	Range start	DECIMAL(15,0)	Range end, rur	DECIMAL(15,2)
Instrument	VARCHAR(128)	Range end	DECIMAL(15,0,)	Range end, rur	DECIMAL(15,2)
Class code	VARCHAR(12)	Discount, %	DECIMAL(15,1)		
Class	VARCHAR(128)	Low rate, rur	DECIMAL(15,6)		
Trading accounts table					
Firm	VARCHAR(12)	Depo account	VARCHAR(12)	Trading account type	VARCHAR(40)

Parameter	Format	Parameter	Format	Parameter	Format
Trading account	VARCHAR(12)	BankAccId	VARCHAR(12)	Depo account type	VARCHAR(30)
Main account	VARCHAR(12)	Description	VARCHAR(32)	Status	VARCHAR(20)
Section type	VARCHAR(12)	T0 settlement organization	VARCHAR(15)	Depo account section	VARCHAR(18)
Uncovered selling prohibition	VARCHAR(12)	T+ settlement organization	VARCHAR(15)		