



QUIK Transactions to XML

Administrator's Manual

Version 3.0



Contents

1. About program	3
1.1 Purpose	3
1.2 Method of operation	3
2. Program setup and configuration.....	4
2.1 System requirements.....	4
2.2 Security requirements.....	4
2.3 Installation	4
2.4 Setup	8
2.5 QT2XML.ini file format	15
2.6 Qcrypto.ini file format	26
2.7 Format of export XML file	27
3. Application finish codes.....	35
4. Transaction statuses.....	36
5. Transaction types.....	37
6. Additional transaction types	37
7. Types of transaction fields.....	37
8. Collaborations between XML file elements and QUIK DB fields	38
9. Modification of created XML file	39
10. Encrypting passwords	40

You may email your comments on this Manual to: quiksupport@arqatech.com



1. About program

1.1 Purpose

QUIK Transactions to XML (qt2xml) program is intended for export of data about transactions from QUIK database to XML file. Data to be exported can be filtered.

1.2 Method of operation

The software is a console application, which is started to export to XML file data about transactions sent by user in a certain time period. Data can be filtered by the following parameters:

- by user identifier (ID);
- by client code;
- by firm code;
- by starting symbols of client code;
- by order number;
- by presence of digital signature;
- by transaction status;
- by transaction type.

As a result of the software work XML file with data about transactions is created.

During work the software can display messages at console, the main of which are:

- **'Exported %n transactions(s)'** – message about quantity of processed transactions, where %n is the quantity of transactions, data about which was exported to the file. The message is displayed after successful Program work finish.
- **'Error: %s'** – error message, where %s is the error description written in a free text format.

After work finishes the program passes a string with code of application finish (errorlevel) to the operational system. This code describes the conditions in which the Program finished its work (for description of application finish codes see [3](#)).

1.2.1 Necessary conditions for Program

For the software to start working access to the QUIK database.



2. Program setup and configuration

2.1 System requirements

For detailed system and software requirements, see [the official website of QUIK](#).

2.2 Security requirements

For the software to start working passwords to access the QUIK database are required to be set. At that the passwords must not begin with symbols \$ and # (if these symbols are used, change them). For security reasons specified passwords cannot be saved to configuration file of the Program in the encrypted form (for details see [10](#)).

2.3 Installation

The commands described below are for Windows OS and Linux OS.

- 1. For Windows OS, the > symbol is indicated at the beginning of the command.**
- 2. For Linux OS, the \$ symbol is indicated at the beginning of the command if the command is executed from the current user, or the # symbol if the command is executed from a user with administrative privileges (usually it is the root user or a user included in the root and wheel groups).**
- 3. The symbols >, \$, and # are not included in the commands.**

2.3.1 Installation on Windows OS

1. Unpack the sent archive qt2xml_<version number>_upd.zip with subdirectories and copy to the target directory.
2. Make the following settings for Windows OS:

For MSSQL DBMS

- When connecting to the MSSQL DBMS, the ODBC driver ODBC Driver 17 for SQL Server must be installed on the computer where qt2xml is installed. To check for driver availability, in the Windows menu, enter the ODBC Data Sources text in the search and open the corresponding window. In the **Drivers** tab, make sure that the driver named ODBC Driver 17 for SQL Server is in the list.

If such driver is missing, download the driver installation file from the official Microsoft website: <https://go.microsoft.com/fwlink/?linkid=2223304> and install it on the computer on which the service will run.



For PostgresPro DBMS

- When connecting to the PostgresPro DBMS, the PostgreSQL ANSI(x64) ODBC driver version no older than 13.2 must be installed on the computer where qt2xml is installed. To check the availability of the driver and its version, in the Windows menu, enter the ODBC Data Sources text in the search and open the corresponding window. In the **Drivers** tab, make sure that the driver named PostgreSQL ANSI(x64) and version 13.2 is listed.

If such driver is missing or the version is older than expected, download the driver installation file from the official postgresql website:

https://ftp.postgresql.org/pub/odbc/versions/msi/psqlodbc_13_02_0000-x64.zip and install it on the computer on which the service will work.

3. Configure the Program settings in the main `qt2xml.ini` file.

- In the `[QUIK_DB]` section, specify the server to connect to the source database:

```
[QUIK_DB]
Server=<Database server name>
```

- Set the remaining parameters in the `[QUIK_DB]` section.

4. Launch the qt2xml Program.

The qt2xml program is used to work in standalone mode as a utility, returning an exit code. For Windows OS, the utility is launched by opening the qt2xml binary file. from the directory of the Program files.

```
> qt2xml
```

During operation, the results of the work are output to the terminal (when launched in the terminal) and written to a log file (by default: qt2xml.log).

When calling the utility, command line arguments can be transmitted to it, for example:

```
> qt2xml -version
```

The -v or -version argument displays the program versions.

After completion of work or if an error occurs, the process will end with a corresponding message in the terminal and/or log file.

2.3.2 Installation on Linux OS

1. Copy the sent `install_qt2xml-<version number>.run` archive to the target directory.
2. Unpack the archive by running the commands:



```
$ chmod +x install_qt2xml-<version number>.run
$ ./install_qt2xml-<version number>.run
```

3. Make the following settings for Linux OS:

- _ To work with ODBC, you need to install the libodbc1 and unixodbc packages using the commands:

```
#apt-get install libodbc1
#apt-get install unixodbc
```

Next, you should configure the necessary drivers for the DBMS.

For MSSQL DBMS

- _ When connecting to the MSSQL DBMS, the ODBC Driver 17 for SQL Server must be installed on the computer where qt2xml is installed.
- _ Run the following commands as a user with administrative privileges:

```
#curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
#curl https://packages.microsoft.com/config/debian/9/prod.list >
/etc/apt/sources.list.d/mssql-release.list
#apt-get update
#ACCEPT_EULA=Y apt-get install -y msodbcsql17
```

- _ Make sure that the following driver entry was added to the /etc/odbcinst.ini file:

```
[ODBC Driver 17 for SQL Server]
Description=Microsoft ODBC Driver 17 for SQL Server
Driver=/opt/microsoft/msodbcsql17/lib64/libmsodbcsql-17.10.so.4.1
```

If the entry was not added, add it yourself.

Make sure the file is located at:

/opt/microsoft/msodbcsql17/lib64/libmsodbcsql-17.10.so.4.1 (file name may vary depending on the version provided by Microsoft.com).

- _ When connecting to a MSSQL Server 2012 DBMS, you need to lower the default encryption level to TLS1.0. This is due to the fact that MSSQL 2012 does not support TLS1.2. To do this, change the values in the /etc/ssl/openssl.cnf file:

```
[system_default_sect]
MinProtocol = TLSv1.0
```



```
CipherString = DEFAULT@SECLEVEL=1
```

This is not required for MSSQL 2017 and newer.

For PostgresPro DBMS

- When connecting to the PostgresPro DBMS, the PostgreSQL ODBC driver version 13.02 or newer must be installed on the computer where qt2xml is installed.
- Run the following commands as a user with administrative privileges:

```
#echo "deb http://apt.postgresql.org/pub/repos/apt buster-pgdg main" >
/etc/apt/sources.list.d/pgdg.list
#wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key
add -
#apt-get update
#apt remove libpq5
#apt remove odbc-postgresql
#apt-get install libpq5/buster-pgdg
#apt-get install odbc-postgresql/buster-pgdg
```

- If installing packages from the official PostgreSQL repository is not available (installation/updating of packages fails), download the packages:
https://apt.postgresql.org/pub/repos/apt/pool/main/p/postgresql-16/libpq5_16.1-1.pgdg100+1_amd64.deb.
https://apt.postgresql.org/pub/repos/apt/pool/main/p/psqlodbc/odbc-postgresql_16.00.0000-1.pgdg100+1_amd64.deb.

- Install the downloaded packages by running the following commands:

```
#apt remove libpq5
#apt remove odbc-postgresql
#dpkg -i libpq5_16.1-1.pgdg100+1_amd64.deb
#dpkg -i odbc-postgresql_16.00.0000-1.pgdg100+1_amd64.deb
```

- Make sure that the following driver entry was added to the /etc/odbcinst.ini file:

```
[PostgreSQL Unicode]
Description=PostgreSQL ODBC driver (Unicode version)
Driver=/usr/lib/x86_64-linux-gnu/odbc/psqlodbcw.so
Setup=libodbcpsqlS.so
Debug=0
CommLog=1
```



```
UsageCount=1
```

If the entry was not added, add it yourself. Make sure the file is located at:
/usr/lib/x86_64-linux-gnu/odbc/psqlodbcw.so (file name may vary depending on version).

If you are working on Windows OS the ANSI driver is used; and when working on Linux OS the UNICODE driver is used.

4. Configure the Program settings in the main `qt2xml.ini` file.

- _ In the `[QUIK_DB]` section, specify the server to connect to the source database:

```
[QUIK_DB]
Server=<Database server name>
```

- _ Set the remaining parameters in the `[QUIK_DB]` section.

5. Launch the qt2xml Program.

The utility is launched by launching qt2xml from the directory of the Program files.

```
./qt2xml
```

Where:

The `./` symbol at the beginning of a command indicates that the command should be executed from the current directory.

During operation, the results of the work are output to the terminal (when launched in the terminal) and written to a log file (by default: `qt2xml.log`).

When calling the utility, command line arguments can be passed to it, for example:

```
./qt2xml -version
```

The `-v` or `-version` argument displays the program versions.

After completion of work or if an error occurs the process will end with a corresponding message in the terminal and/or log file.

2.4 Setup

The software is a console application. After start the program reads settings from the configuration file (for more details about settings from the configuration file see 2.5). Besides configuration file parameters can be setup from the command line. If one option is setup both in the configuration file and by command line then value from the command line has the priority.



qt2xml software supports the following keys:

- **-period<period start>-<period finish>,<time interval start>-<time interval finish>**
– start and finish dates of period for the transaction data export are setup by <YYYYMMDD> (date) format or <YYYYMMDDHHMMSS> (date and time) format. Start and finish dates of period can be setup by different formats. Only date and time of period start are included in report. If interval boundary has only date then it is supposed that day starts at 00:00:00 and finishes when the next day starts.
Start and finish time of interval for transactions data export are setup by <HHMMSS> (time) format. If the time interval is set up the transaction data is exported every day during the specified period in this time interval. If the start and/or finish time of interval is out of the specified period on the first or last day of this period the transaction data is exported only within the start and/or finish period time. If the key is not setup and -periodprevday key is not used then current day is used as a period.

For example:

```
qt2xml -period20090501-20090501183000,140000-180000
```

- **-periodprevday<time interval start>-<time interval finish>** – specified time interval of the previous day is selected as a period. Start and finish time of interval are setup by <HHMMSS> format. If the interval is not set up the previous day is used as a period.
For example:

```
qt2xml -periodprevday
```

- **-uid<user ID>** – filter of data about transactions by ID of users. User IDs in the list are separated by comma. If the key is not setup then data about all users is exported.
For example:

```
qt2xml -uid6, 12,418
```

- **-excludeuid** – excluding transactions data filter on ID of users. User IDs in the list are separated by comma. The data is transmitted for all users except of those specified in key.

For example:

```
qt2xml -excludeuid6,12,418
```



If both filters `-uid<user ID >` and `-excludeuid` are set then the excluding filter `-excludeuid` is ignored.

- **`-client<client code or mask word>`** – filter of data about transactions by client code or mask word (starting symbols of client code and asterisk (*) are typed). If the key is not setup then data about all client codes is exported.

For example:

```
qt2xml -clientE100 or qt2xml -clientE*
```

- **`-nouserinfo`** – prohibition of UserInfo element export.
- **`-status<status symbol>`** – filter of data about transactions by transaction status (each status is marked by some symbol, see 4). The key value can include one or several transaction statuses. The statuses are listed in any order without any separators. If the key is not setup then data to be exported does not depend on transaction status.

For example:

```
qt2xml -statusof
```

- **`-output<file name>`** – name of XML file for data export. If the key is not setup then data about transactions is exported to trans.xml file which is created automatically and is situated in the current directory.

For example:

```
qt2xml -outputTransReport.xml
```

- **`-ini<file name>`** – absolute or relative file path to the configuration file. If the key is not setup then the program looks for qt2xml.ini file in the current directory.

For example:

```
qt2xml -ini.\qt2xml.ini
```

- **`-classes<list>`** – list of security classes which are available for export of data about transactions separated by commas.

For example:

```
qt2xml -classesEQBR,EQNE
```



- **-excludeclasses<list>** – list of security classes which are prohibited for export of data about transactions separated by commas.

For example:

```
qt2xml -excludeclassesPSEQ,PSES
```

If -classes and -excludeclasses lists are not setup then data about all classes is exported. If both lists are setup then -excludeclasses list is ignored.

- **-securities<list>** – list of available security codes separated by commas.

For example:

```
qt2xml -securitiesRTKM,HYDR
```

- **-excludesecurities<list>** – list of security codes to be ignored separated by commas.

For example:

```
qt2xml -excludesecuritiesBUSB,YKEN
```

If -securities and -excludesecurities lists are not setup then data about all securities is exported. If both lists are setup then -excludesecurities list is ignored.

- **-reply<text>** – substring to be looked for in the reply for transaction. Any string can be used as <text>. If the string has tabs then it must be written with double quotes.

For example:

```
qt2xml -reply"Operations with current trade account are prohibited for you"
```

- **-encodenonchar** – change export format of data about transaction and its fields if the transaction has non-printable symbols (besides symbol with 0x20 code (tab)).

For example:

```
qt2xml -encodenonchar
```

- **-checksign** – check availability of digital signature for each transaction.

For example:



```
qt2xml -checksign
```

1. Before checking availability of digital signature for transactions make sure that QT2XML catalog includes `qcrypto.ini` file with crypto provider settings (see 2.6) and `CryptoType` parameter of configuration file has the type of crypto provider which is used (see 2.5.3).
2. If checking for digital signature is enabled, select the parameter `cert_storage_validation=0` in section `[signal_com_message_pro_common_settings]` of `signalcom_pr.ini` file when using the SignalCom crypto provider.

- **-signoutputfile** – create digital signature for XML file.

For example:

```
qt2xml -signoutputfile
```

- **-withsign** – export transactions with digital signature only.

For example:

```
qt2xml -withsign
```

- **-withnosign** – export transactions without digital signature only.

For example:

```
qt2xml -withnosign
```

If both -withsign and -withnosign keys are setup then -withnosign key is ignored.

- **-transflags<set of flags>,<filter type>** – filter by Flags field in transaction description (each value is setup by a certain set of bits, see 2.7.4), where
 - **<set of flags>** are flags which must be included in transaction description. Set of flags is specified by bit mask or by listing flags comma separated.
 - **<filter type>** is setup according to the following rule:
 - any – transaction may have any of listed flags;
 - exact – only full collaboration of listed flags is allowed;
 - all (by default) – all the listed flags are allowed.



Tabs in the filter are not allowed.

Examples:

Set of flags is defined by bit mask:

```
qt2xml -transflags3,any
```

Set of flags are listed:

```
qt2xml -transflags1024,2048,all
```

- **-excludetransflags<set of flags>,<filter type>** – filter by Flags field in transaction description (each value is setup by a certain set of bits, see 2.7.4), where:
 - **<set of flags>** are flags which must not be included in transaction description. Set of flags is specified by bit mask or by listing flags comma separated;
 - **<filter type>** is setup according to the following rule:
 - any – transaction must not have any of flags listed in filter;
 - exact – only full collaboration of listed flags is prohibited;
 - all – all flags listed in filter are prohibited.

Tabs in the filter are not allowed.

For example:

```
qt2xml -excludetransflags1024,any
```

- **-firms<list>** – list of available firm codes. Firm codes in the list are separated by commas.
For example:

```
qt2xml -firmsMC0080100000,NC0058900000
```

- **-excludefirms<list>** – list of firm codes to be ignored. Firm codes in the list are separated by commas.
For example:

```
qt2xml -excludefirmsNC0038900000
```



- **-ordernum<number_1>[,...<number_n>]** – filter by orders/stop-orders numbers in transaction. Order / stop-order numbers in the list are separated by commas.
For example:

```
qt2xml - ordernum5487,5641
```

- **-exchangenum<number_1>[,...<number_n>]** – filter for exchange numbers of orders, non-trade instructions for transaction.

1. **Commas (,) and the quotation marks (") cannot be used in the exchange number.**
2. **If at least one number contains a gap, the entire list of numbers must be enclosed in quotation marks ("):**

```
-exchangenum"<number_1>[,...<number_n>]"
```

For example:

```
qt2xml -exchangenumML0125,ML 0132"
```

In some keys filter parameters may not be setup. In this case filter from configuration file is deactivated. This rule is applied to the following keys: -period, -uid, -excludeuid, -client, -status, -reply, -classes, -excludeclasses, -transflags, -excludetransflags, -firms, -excludefirms.

- **-exportmode<mode>** – mode of exporting data on transactions.
For example:

```
qt2xml -exportmode1
```

- **-transnum<number_1>[,...<number_n>]** – filter by transaction numbers. Transaction numbers in the list are separated by commas.
For example:

```
qt2xml -transnum641
```



2.5 QT2XML.ini file format

2.5.1 [QUIK_DB] section

[QUIK_DB] section contents settings of connection to the QUIK database.

- **AuthMode** – authentication type. Valid values:
 - _ login (by default) – authentication via login and password specified in parameters **Login** and **Password**;
 - _ domain – using domain authentication, parameters **Login** and **Password** are ignored.
- **Type** – type of database to connect to. Valid values:
 - _ MSSQL (by default);
 - _ POSTGRES.
- **Driver** – name of ODBC driver, used for connection to database. Valid values:
 - _ **ODBC Driver 17 for SQL Server** (by default) – used when working with MS SQL Server 2014, 2016 or newer DBMS versions;
 - _ **PostgreSQL ANSI(x64)** – used when working with PostgreSQL on Windows;
 - _ **PostgreSQL Unicode** – used when working with PostgreSQL on Linux.
- **SSLMode** – mode of encrypting when connected to PostgreSQL server. Valid values:
 - _ **require** – SSL encryption (if the server refuses encryption, then a shutdown is performed);
 - _ **verify-full** – SSL encryption with validation on both sides.

If there is no setting, the connection is made without encryption.
- **CertCAFileName** – path to root certificate file.
- **CertFileName** – path to user certificate file, if certificate authentication is used.
- **CertPKeyFileName** – path to private key of user's certificate file.
- **CertPass** – user's password. The password must not begin with "\$" and "#" symbols. Passwords can be encrypted (for details see [10](#)).
- **AdditionalParameters** – additional parameters of connection to data base.

For example:

```
[QUIK_DB]
CertCAFileName=Certs/root.crt
CertFileName=Certs/user.crt
CertPKeyFileName=Certs/user.key
CertsPass=qwe
```



```
AdditionalParameter=MultiSubnetFailover=true <MultiSubnetFailover=true is supported  
ODBC Driver 17 for SQL Server and later>
```

For direct connection to SQL server the section must content the following keys:

- **Server** – SQL server name;
- **DataBase** – name of server database;
- **User** – login for work with database;
- **Password** – password for work with database. The password must not begin with "\$" and "#" symbols. The password can be encrypted (for details see [10](#)).

For example:

```
[QUIK_DB]  
Server=QDB1  
DataBase=AQ_SERV1  
User=sa  
Password=qwe
```

For connection to database via ODBC (for example if Interbase DBMS is used) the section must content the following keys:

- **DSN** – name of ODBC source which points to Interbase database;
- **User** – login for work with database (if needed);
- **Password** – password for work with database (if needed). The password must not begin with "\$" and "#" symbols. The password can be encrypted (for details see [10](#)).

For example:

```
[QUIK_DB]  
DSN=SERV1  
User=sa  
Password=qwe
```

2.5.2 [General] section

[General] section contents keys values of which are the same with corresponding parameters of command line:

- **UseXSL** – name of the file with style used for viewing the content of XML-file (for more details on modification of XML-file see [9](#)).



- **Period** – setting format is the following: **<period start>-<period finish>,<time interval start>-<time interval finish>** where start and finish dates of period for data about transactions export are setup by <YYYYMMDD> (date) format or <YYYYMMDDHHMMSS> (date and time) format. Start and finish dates of period can be setup by different formats. Only date and time of period start are included in report. If interval boundary has only date then it is supposed that day starts at 00:00:00 and finishes when the following day starts. Start and finish time of interval for transactions data export are setup by <HHMMSS> (time) format. If the time interval is set up the transaction data is exported every day during the specified period in this time interval. If the start and/or finish time of interval is out of the specified period on the first or last day of this period the transaction data is exported only within the start and/or finish period time. If the key is not setup then current day is used as a period.

For example:

```
Period=20090602101500-20090602183000,140000-180000
```

- **UID** – filter of data about transactions by ID of users. User IDs in the list are separated by commas. If the key is not setup then data about all users is exported.

For example:

```
UID=6
```

- **ExcludeUID** – excluding transactions data filter on ID of users. User IDs in the list are separated by commas. The data is transmitted for all users except of those specified.

For example:

```
UID=6,12,418
```

If both filters UID and ExcludeUID are set, then the excluding ExcludeUID filter is ignored.

- **Client** – filter of data about transactions by client code or mask word (starting symbols of client code and asterisk (*) are typed).

For example:

```
Client=D123
```



- **NoUserInfo** – control UserInfo element export. Valid values:

- _ 0 – the element is exported;
- _ 1 – the element is not exported.

For example:

```
NoUserInfo=1
```

- **Status** – filter of data about transactions by transaction status (each status is marked by some symbol, see 4). The key value can include one or several transaction statuses. The statuses are listed in any order without separators. If the key is not setup then data to be exported does not depend on transaction status.

For example:

```
Status=rg
```

- **Reply** – data filter by substring in QUIK server reply for transaction. If the filter is not setup then data to be exported does not depend on reply for transaction.

For example:

```
Reply=Operations with current trade account are prohibited for you
```

- **Output** – name of file for data export. Depending on mode of data export selected in parameter **ExportMode** value of the parameter takes the following forms:
 - _ If ExportMode=1 name of XML file and its path are specified when necessary. If the key is not setup then data about transactions is exported to trans.xml file, which is created automatically and is situated in the current directory;
 - _ If ExportMode=2 or ExportMode=3 path (when necessary) and template of the file name are specified in format '**<Path>\<Template>**', where: **<Template>** – beginning of the file name. During the export the file name takes format '**<Path>\<Template><Number of transaction>.<file extension>**', where **<Number of transaction>** – number of transaction supplemented at the left by zeros up to length of 10 characters (for example, '0001234567').

For example:

```
Output=TransReport.xml
```



- **Classes** – list of security classes which are available for export of data about transactions separated by commas. The parameter may stay empty.

For example:

```
Classes=EQBR,EQNE
```

- **ExcludeClasses** – list of security classes which are prohibited for export of data about transactions separated by commas. The parameter may stay empty.

For example:

```
ExcludeClasses=PSEQ,PSES
```

If Classes and Excludeclasses lists are not setup then data about all classes is exported. If both lists are setup then Excludeclasses list is ignored.

- **EncodeNonChar** – change export format of data about transaction and its fields if the transaction has non-printable symbols (besides symbol with 0x20 code (tab)). Valid values:

- _ 0 (by default)– the format is not changed;
- _ 1 – the format is changed.

For example:

```
EncodeNonChar=1
```

- **CheckSign** – check availability of digital signature for each transaction. Valid values:

- _ 0 – digital signature is not checked (by default);
- _ 1 – digital signature is checked.

If there is digital signature, then the digital signature correctness is checked for the current transaction and key (certificate) is identified, with the help of which the digital signature was created.

For example:

```
CheckSign=1
```



- **SignOutputFile** – create digital signature for XML file based on settings from [FileSignSettings] section (for more details see [2.5.4](#)). Valid values:
 - _ 0 (by default)– file is not signed;
 - _ 1 – file is signed.

For example:

```
SignOutputFile=1
```

- **Withsign** – export transactions with digital signature only. Valid values:
 - _ 0 (by default)– all transactions are exported;
 - _ 1 – transactions with digital signature only are exported.

For example:

```
Withsign=1
```

- **Withnosign** – export transactions without digital signature only. Valid values:
 - _ 0 (by default)– all transactions are exported;
 - _ 1 – transactions without digital signature only are exported.

For example:

```
Withnosign=0
```

If both 'withsign' and 'withnosign' parameters are setup then only 'withsign' parameter is used.

- **TransFlags** – filter by Flags field in transaction description (each value is setup by a certain set of bits, see [2.7.4](#)) in format:

```
TransFlags=<setoff flags>[,<filter type>]
```

Where:

- _ **<set of flags>** are flags which must be included in transaction description. Flags are specified by bit mask or listed comma separated.
- _ **<filter type>** is setup according to the following rule:



- _ any – transaction may have any of flags listed in filter;
- _ exact – only full collaboration of listed flags is allowed;
- _ all (by default) – all flags listed in filter are required.

For example:

Set of flags is defined by bit mask:

```
TransFlags=3,any
```

Flags are listed:

```
TransFlags=8192,134217728,2147483648,any
```

- **ExcludeTransFlags** – filter by **Flags** field in transaction description (each value is setup by a certain set of bits, see 2.7.4) in format:

```
ExcludeTransFlags=<set of flags>[,<filter type>]
```

Where:

- _ **<set of flags>** are flags which must be included in transaction description. Set of flags is specified by bit mask or by listing flags comma separated.
- _ **<filter type>** is setup according to the following rule:
 - _ any – transaction must not have any of flags listed in filter;
 - _ exact – only full collaboration of listed flags is not allowed;
 - _ all (by default) – all flags listed in filter are not allowed.

For example:

Set of flags is defined by bit mask:

```
ExcludeTransFlags=1024,any
```

Flags are listed:

```
ExcludeTransFlags=512,1024,any
```



- **BaseStopOrder** – control export of **BaseStopOrder** and **BaseStopOrderTime** parameters of basic stop order which created the current stop order. Valid values:
 - _ 0 (by default)– do not export;
 - _ 1 – export.

For example:

```
BaseStopOrder=0
```

- **Securities** – list of available security codes separated by commas.

For example:

```
Securities=RTKM,HYDR
```

- **ExcludeSecurities** – list of security codes to be ignored separated by commas.

For example:

```
ExcludeSecurities=USB,YKEN
```

If Securities and Excludesecurities lists are not setup then data about all securities is exported. If both lists are setup then Excludesecurities list is ignored.

- **Firms** – list of available firm codes separated by commas.

For example:

```
Firms= MC0080100000,NC0058900000
```

- **ExcludeFirms** – list of firm codes to be ignored separated by commas.

For example:

```
ExcludeFirms=NC0038900000
```

If Firms and ExcludeFirms lists are not setup then data about all firms is exported. If both lists are setup then Excludefirms list is ignored.



- **OrderNum** – filter by orders/stop-orders numbers in transaction. Order/stop-order numbers in the list are separated by commas.

For example:

```
OrderNum=1203,1209,123456
```

- **ExchangeNum** – filter for exchange numbers of orders, non-trade instructions for transaction in the form of a random string.

For example:

```
ExchangeNum=987654,20150725 1234567,20150725 5678901
```

- **StopOrderExtraInfo** – the setting to manage displaying the additional information on stop orders. Valid values:

- _ 0 (by default) – do not display;
- _ 1 – display.

The following attributes are displayed additionally upon the enabled setting:

- _ ResultOrderNum – number of the placed order;
- _ LinkedOrder – number of linked order;
- _ GeneratedTransNum – internal number of transaction to place order.

For example:

```
StopOrderExtraInfo=1
```

- **ExportMode** – mode of exporting data on transactions. Valid values:
 - _ 1 (by default) – export data in XML format;
 - _ 2 – export data in a file with extension .sgn. Transaction body and the digital signature (if available) are saved;
 - _ 3 – export data in three files. Separate files with different extensions save:
 - _ .data – transaction body;
 - _ .sign – digital signature;
 - _ .desc – description of the transaction fields.

Path to files of exporting data is defined by parameter [Output](#).



In export modes 2 and 3 parameters UseXSL, NoUserInfo, EncodeNonChar, CheckSign, SignOutputFile, BaseStopOrder, WebQuik, StopOrderExtraInfo (and the same name command-line parameters) are ignored.

- **TransNum** – filter by transaction number. Transaction numbers in the list are separated by commas.

For example:

```
TransNum=641,679
```

- **TransInfoType** – filter by transaction type (each transaction type is specified by a code, see section 5). Transaction types in the list are separated by commas.

For example:

```
TransInfoType=67,91
```

- **ExcludeTransInfoType** – excluding filter by transaction type (each transaction type is specified by a code, see section 5). Transaction types in the list are separated by commas.

For example:

```
ExcludeTransInfoType=67,91
```

If both filters TransInfoType and ExcludeTransInfoType are specified, then the excluding filter ExcludeTransInfoType is ignored.

- **TransInfoExType** – filter by additional transaction type (each transaction type is specified by a code, see section 6). Additional transaction types in the list are separated by commas.

For example:

```
TransInfoExType=3,4
```

- **ExcludeTransInfoExType** – excluding filter by additional transaction type (each transaction type is specified by a code, see section 6). Additional transaction types in the list are separated by commas.

For example:



```
ExcludeTransInfoExType=3,4
```

If both filters `TransInfoExType` and `ExcludeTransInfoExType` are specified, then the excluding filter `ExcludeTransInfoExType` is ignored.

- **Language** defines language used when exporting descriptions of transactions and their fields. Value is not case sensitive. Valid values:

- _ ru (by default) – Russian;
- _ en – English.

If a transaction description or its field is not available in the selected language, description in other language is used.

- **DebugLevel** – level of the application events logging. Valid values: from 0 to 250, where:
 - _ 0 – events are not logged;
 - _ 1 (by default) – the highest priority events are logged;
 - _ 250 – all events are logged.

2.5.3 [SignSettings] section

If the mode of checking digital signature is used then the type of crypto provider should be specified in the current section. Settings of the crypto provider are stored in `qcrypto.ini` file in the current catalogue.

- **CryptoType** – crypto provider type. Valid values:
 - _ SignalCom – for Linux OS, only this crypto provider is used. For Windows OS it is used only for compatibility purposes, in fact the **SSLAndMessagePro** is in use;
 - _ SSLAndMessagePro;
 - _ BiCrypt;
 - _ CryptoPro;
 - _ MessagePro

For example:

```
CryptoType=SSLAndMessagePro
```



2.5.4 [FileSignSettings] section

This section has settings of crypto provider which is used for creating digital signature for XML file.

- **CryptoType** – crypto provider type. Valid values:
 - _ SignalCom – for Linux OS, only this crypto provider is used. For Windows OS it is used only for compatibility purposes, in fact the **SSLAndMessagePro** is in use;
 - _ SSLAndMessagePro;
 - _ BiCrypt;
 - _ CryptoPro;
 - _ MessagePro.

For example:

```
CryptoType=BiCrypt
```

If the both modes of checking digital signature and creating digital signature for XML file are used then crypto provider settings in [SignSettings] and [FileSignSettings] sections must be the same.

2.6 Qcrypto.ini file format

Crypto provider must be configured if CheckSign=1 and SignOutputFile=1.

2.6.1 [CryptoProviders] section

The section includes settings of the following view:

- **<type of crypto provider>** – attribute that indicates using crypto provider <type of crypto provider>. Valid values:
 - _ 0 – is not used;
 - _ 1 – used.

2.6.2 [CryptoProvidersSettings] section

- **CipherCryptoProvider** – current type of crypto provider.
- **SignCryptoProvider** – name of current crypto provider.



2.6.3 Sections of [<type of crypto provider>] view

- **module** – path to crypto provider’s library file.
- **IniFile** – path to crypto provider’s configuration file.

2.6.4 File example

```
[CryptoProviders]
BiCrypt=0
SignalComMessageAndSSLPro=0
CryptoPro=1
MessagePro=0

[CryptoProvidersSettings]
SignCryptoProvider=CryptoPro
CipherCryptoProvider=CryptoPro

[MessagePro]
Module=.\signalcom_pr.dll
IniFile=.\signalcom_pr.ini

[SignalComMessageAndSSLPro]
Module=.\signalcom_pr.dll
IniFile=.\signalcom_pr.ini

[CryptoPro]
Module=.\CPCSP_Pr.dll
IniFile=.\cryptoCSP_Pr.ini

[BiCrypt]
Module=.\bicrypt_pr.dll
IniFile=.\bicrypt_pr.ini
```

2.7 Format of export XML file

For creation of XML file version 1.0 is used. If you are working on Windows the-1251 encoding file is used; when working on Linux the UTF-8 coding file is used. In export files all the fields and data types belong to the “urn:quik:trans-info:v1.0” name space.

The file has the root node of **TransactionsReport** type (for more details about attributes of TransactionsReport root node see [2.7.1](#)). There can be any quantity of nodes of **Trans** type inside the root node. Each of these nodes describes a separate transaction which was sent by user (for more details about attributes of **Trans** node see [2.7.2](#)). Node of **Trans** type contents the following elements:



- **UserInfo** – if this element exists then it describes user who sent the transaction (for more details about attributes of **UserInfo** element see 2.7.3).
- **Data** – contents transaction text in string format. Tabs are significant characters.
- **DataEncoded** – contents transaction text in encoded base64 format.

If EncodeNonChar option is activated (see 2.5.2) then an empty string is recorded to Data element, and transaction text is recorded to DataEncoded element.

If EncodeNonChar option is deactivated then DataEncoded element is missed.

- **PureData** – if this element exists then it contents the full text of transaction including digital signature and situations when the digital signature is available in transaction but its check is not configured on server. The content of transaction is encoded by base64 algorithm. Tabs and line breaks are ignored. It is possible that digital signature is not used, but its text is present.
- **Reply** – if this element exists then it contents trading system reply for transaction. Tabs are significant characters.
- **Sign** – if this element exists then it contents text of digital signature received from the client. The content of digital signature is encoded by base64 algorithm. Tabs and line breaks are ignored. If the element is missed then it means that transaction does not have a received text of digital signature.
- **TransData** – this node contents description of this type transaction and values of all its fields. The parameter contents attributes which describe the transaction (for more details about attributes of **TransData** element see 2.7.4). There are elements of **Field** type inside the node of **TransData** type. They describe separate transaction fields (for more details about attributes of Field element see 2.7.5).
- **BaseStopOrder** – if this element exists then it contents stop order number on the bases of which current transaction was created.
- **BaseStopOrderTime** – if this element exists then it contents date and time of stop order registration within the accuracy of microseconds, on the bases of which current transaction was created.

2.7.1 Attributes of TransactionsReport node

Attribute name	Type	Description
StartDate	date	Start date of data export
EndDate	date	End date of data export
UID	integer	If the element exists then the file includes only transactions which were sent by user with the selected ID
ExcludeUID	string	If the element is present then the file includes all transactions instead of those sent by the user with the specified ID
ClientCode	string	If the element exists then the file includes only transactions which were



Attribute name	Type	Description
		sent by user with the selected client code. If the value of element has asterisk (*) in the end then the file includes transactions which were sent by users with client codes starting from the selected symbols (* is treated as any substring)
TransStatus	string	If the element exists then the file includes only transactions with status which is the same with selected one
TransFlags	string	Filter type in format: <list type>, <flags>, <filter type>, where all the parameters are typed one after another (without tabs) and have the following description: – <list type>: – 1 - available transactions are setup; – 0 - prohibited transactions are setup; – <flags> – flags from filter (in hexadecimal format without leading nulls, without prefix 0x but with postfix h); – <filter type> – one of values all, any or exact (depending on what is setup in the filter)
SignAvailable	integer	Attribute of transaction export with/without digital signature. Valid values: – 0 – transactions without digital signature only are exported; – 1 – transactions with digital signature only are exported
OrderNum	int64	If the element exists then the file includes only transactions with selected order number
ExchangeNum	string	If the element is present then the file includes all transactions with specified exchange number of the order, non-trade instruction
TransInfoType	string	If the element is specified, then only the transactions, which fit the filter by transaction type, are included in the file. The filter is specified in format <list type>,<transaction type> [,<transaction type>], where all the parameters are specified one by one without tabs and have the following description: – <list type>: – 1 – the allowed transaction types are specified; – 0 – the prohibited transaction types are specified; – <transaction type> – the value of transaction type
TransInfoExType	string	If the element is specified, then only the transactions, which fit the filter by additional transaction type, are included in the file. The filter format is the same with the format of filter specified by the TransInfoExType attribute

2.7.2 Attributes of Trans node

Attribute name	Type	Description
TransNum	integer	Inside number of transaction in QUIK DB
UID	integer	User ID who sent the transaction



Attribute name	Type	Description
TransID	integer	Transaction ID in the current session
SessionID	integer	Session ID in which the transaction was sent
TradeDate	datetime	Date of trade day
Status	string	Transaction status
QuikDate	datetime	Date of transaction registration on QUIK server
QuikTime	datetime	Time within the accuracy of microseconds, when QUIK server received the transaction
ReplyTime	datetime	Time within the accuracy of microseconds, when QUIK server received reply for the transaction from trading system
OrderNum	int	If the element exists then it contains number in trading system which was assigned to the order sent by the transaction
ExchangeNum	string	If the element is present then it includes exchange number of the order assigned on the western market, or number of non-trade instruction
ClientCode	string	If the element exists then it has client code from the transaction
SignStatus	int	Status of digital signature check. Valid values: 0 – digital signature is used. Received text of the digital signature is correct. Data about key (certificate) is in SignerCertSerialNumber, SignerCertSubject, SignerCertIssuer, SignerCertCryptoType attributes; 1 – digital signature is present in transaction, its check is configured on server but the digital signature failed the check. SignErrorMsg attribute has the text of error. Data on key (certificate) is specified in attributes SignerCertSerialNumber, SignerCertSubject, SignerCertIssuer, SignerCertCryptoType; 2 – digital signature is present in transaction, but its check is configured on server. The Sign element contains the text of digital signature received from the client. If digital signature is not present in transaction, and its check is not configured on server, the SignStatus attribute is not displayed
SignerCertSerialNumber	string	Serial number of the key (certificate) by which digital signature for transaction was created. If digital signature is not correct or serial number of the key (certificate) is not received then the attribute is not used. The element exists if digital signature is used and mode of digital signature check is activated. If any data on the certificate is not found, value 'Unavailable' will be displayed

Attribute name	Type	Description
SignerCertSubject	string	Data about owner of the key (certificate) by which digital signature for transaction was created. If digital signature is not correct or data about key (certificate) owner is not received then the attribute is not used. The element exists if digital signature is used and mode of digital signature check is activated. If any data on the certificate is not found, value 'Unavailable' will be displayed
SignerCertIssuer	string	Data about certification authority of the key (certificate) by which digital signature for transaction was created. If digital signature is not correct or data about key (certificate) certification authority is not received then the attribute is not used. The element exists if digital signature is used and mode of digital signature check is activated. If any data on the certificate is not found, value 'Unavailable' will be displayed
SignerCertCryptoType	string	Certificate type. Available only if the digital signature is used. If any data on the certificate is not found, value 'Unavailable' will be displayed
*SignErrorMsg	string	Error message received from crypto provider. The parameter exists if error arose while digital signature check (including situation when data about the key by which digital signature was created was not found)
LinkedOrder	integer	Number of linked order
ParentTransID	integer	Base transaction ID for additional restrictions

* – an error message received from crypto provider is corrected as follows: unprintable characters are deleted, line breaks are replaced with spaces, the message text is truncated to 256 characters.

2.7.3 Attributes of UserInfo element

Attribute name	Type	Description
Name1	string	User name
Name2	string	User middle name
Name3	string	User surname
OrgCode	integer	Firm code to which the user refers
OrgName	string	Firm name to which the user refers
Login	string	User login in other system. The parameter is used for authorization by RSA SecurID or Active Directory facilities



2.7.4 Attributes of TransData element

Attribute name	Type	Description
ClassID	integer	ID of security class to which the transaction refers
ClassCode	string	Code of security class to which the transaction refers
InfoType	integer	Transaction type
InfoExType	integer	Additional transaction type. Is used for a more concrete identification of transaction type
Description	string	Transaction description (name)
FieldsNum	integer	Quantity of fields in transaction
Flags	integer	Transaction flags (internal QUIK flags). The field has a view of bit mask. For values of bits see the following table
ClassType	integer	Class type. Valid values: – 1 – Securities; – 2 – Bonds; – 3 – Futures; – 4 – Options; – 5 – Spreads; – 6 – Swaps; – 7 – Currency; – 8 – Strategies; – 9 – Indices; – 11 – Reports; – 12 – Instructions; – 13 – Autochartist; – 14 – Autofollow; – 16 – Algorithmic orders; – 17 – SMS messages; – 18 – Currency cross rates; – 21 – OTC trades; – 22 – Precious metals; – 23 – QUIK Service Data; – 24 – OTC market; – 25 – Deposits; – 26 – Investment units; – 27 – Messaging; – Otherwise – unknown type
ClassSubType	integer	Class subtype. Valid values: – Algorithmic orders: – 0 – not specified; – 1 – Iceberg; – 2 – Volatility; – 3 – TWAP; – 4 – VWAP; – 5 – GTC/GTD orders; – 6 – Stop order; – 7 – Spread; – 8 – unknown type. – Swaps: – 0 – not specified; – 1 – Standard currency SWAP; – 2 – Short currency SWAP (one day);



Attribute name	Type	Description
		– 3 – Forward currency SWAP; – 4 – unknown type. – Spreads: – 0 – not specified; – 1 – Stock spread; – 2 – Futures spread; – 3 – Options spread; – 4 – Currency spread; – 5 – unknown type. – QUIK Service Data: – 4 – Information about clients; – 5 – Dictionaries. – Messaging: – 1 – Broadcast channels; – 2 – Support service; – 3 – Support group members
ClassBehavior	integer	Class features. Valid values: – 1 – Permissions by parameters; – 2 – Do not show redemption; – 4 – SOR; – 8 – Active operations; – 16 – Estimate return in quotes; – 32 – Additional synchronization; – 128 – REPO class; – 25 – NDM class; – 4096 – Modified REPO class; – 134217728 – OMS orders; – 64 – REPO with CCP; – 4294967296 – External algo orders; – 8589934592 – NDM with CCP; – 1024 – Informational class; – 8192 – Class of participation in offerings
ClassSpecialBehavior	integer	Special class features. Valid values: – 1 – Amount in the currency; – 2 – Fractional quantity

Bit flag values of Flags field of transaction:

Flag	Description	Flag	Description
0	Additional restrictions control	128	New report for execution
1	Remove order	256	Remove report for execution
2	Remove OTC order	512	Setup futures limit
4	New order	1024	Remove futures limit
8	New OTC order	2048	Remove all active orders
64	Noncompetitive order for sending	8192, 134217728 or 2147483648	New stop order



Flag	Description
16384	Remove stop order
32768	New REPO order
65536	Order for money output
131072	Order for securities output
262144	New non-addressed order
524288	New REPO non-addressed order

Flag	Description
1048576	Remove non-addressed order
8388608	Setup futures limit for the firm
8388612	New algo order
16777216	Assign for stop order
67108864	Activate stop order

Bit masks of Flags parameter can consist of several flags. That is why bit values must be checked if they are included in the value and must not be checked if they are equal to the value.

2.7.5 Attributes of Field element

Attribute name	Type	Description
Number	integer	Number of transaction field
Name	string	Name of transaction field
Description	string	Description of transaction field
Size	integer	Size of transaction field
Type	integer	Type of transaction field (see 7)
Flags	integer	Flags of transaction field (internal QUIK flags)
DefValue	string	Value of transaction field by default
EnumCount	integer	Quantity of specified type values in the field
EnumValues	string	Description of specified type values for the current field. The attribute contents the algorithm of transformation of specified type values into strings
Value	string	Values of transaction field for the current transaction. Attribute stays empty if EncodeNonChar mode is activated and transaction field has unprintable symbols
PreparedValue	string	Transformed value of transaction field: the quantity of values after point for numbers with floating point is taken into account, values of specified type are transformed into strings. Attribute stays empty if EncodeNonChar mode is activated and transaction field has unprintable symbols



Attribute name	Type	Description
ValueEncoded	string	Values of transaction field for the current transaction encoded by base64 algorithm. If the field does not have any unprintable symbols then ValueEncoded is not used
LotSize	integer	Size of security lot
Scale	integer	Scale used for the field. Parameter for fields of 0, 2, 3, 6 types (see 7)
DateType	integer	Type of specified date. Valid values: – default – date is not set; – today – field contains date of the current day; – GTC – field contains 'Till Cancel' value; – date – field contains date value from attribute PreparedValue. Used only for fields of 5 type (see 7)
InfoType	integer	Type of transaction field
InfoExType	integer	Additional type of transaction field. The attribute is used for more accurate field type identification

3. Application finish codes

Return code

(errorlevel) Description

0	Errors while software work were not found, transactions were exported to the file
1	Errors while software work were not found, transactions which meet the conditions were not found
2	Access to database denied
3	Error of data in transactions table (including error in session description, error in transaction description or transaction field description)
4	Error of XML file creation/record
5	Incorrect input data
6	Other errors



4. Transaction statuses

Flag	Value	Transaction status
TRANS_STATUS_RECEIVED	r	Is received by QUIK server
TRANS_STATUS_GATE_FAULT	g	Error of connection with trading system
TRANS_STATUS_TRADE_OK	o	Transaction was executed successfully by trading system
TRANS_STATUS_TRADE_FAULT	f	Trading system returned an error
TRANS_STATUS_QUIK_CHECK_FAULT	q	Transaction did not pass check at QUIK server
TRANS_STATUS_LIMIT_CHECK_FAULT	l	Transaction did not pass limit control
TRANS_STATUS_ACCEPTED	a	Transaction has been confirmed by manager
TRANS_STATUS_REJECTED	j	Transaction has been rejected by manager
TRANS_STATUS_KILLED	k	Transaction has been cancelled by user
TRANS_STATUS_NOT_SUPPORTED	u	Transaction is not supported by server
TRANS_STATUS_VERBA_FAULT	v	Error of digital signature
TRANS_STATUS_CROSS_TRADE_REJECTED	c	Transaction has been declined because of cross trade
TRANS_STATUS_TRANS_COLLECT_FAILED	p	Transaction has not been received fully
TRANS_STATUS_SOFT_LIMIT_CHECK_FAULT	s	Transaction has not passed the additional restrictions control
TRANS_STATUS_ACCEPTED_AFTER_SOFT_LIMIT_CHECK_FAULT	b	At first transaction has not passed the additional restrictions control, executed
TRANS_STATUS_STOPPED_AFTER_SOFT_LIMIT_CHECK	i	At first transaction has not passed the additional restrictions control, cancelled



5. Transaction types

Value	Transaction type
1	Stop-order entry
67	Order withdrawal
91	Specify cash limit at derivatives market
133	Entry of address REPO order

6. Additional transaction types

Value	Additional transaction type
3	Stop-order
4	Broker stop-order
10002	Order from WSE
10005	Order from LSE

7. Types of transaction fields

Type of transaction field	Description
0	String of symbols. Size of symbol string is defined by size of transaction field
1	Integer number written in view of string, can be written with some scale (scale is defined separately)
2, 3	Real number written in view of string, can be written with some scale (scale is defined separately)
4	Specified type. If value of ENUM_COUNT is 0 then field content is taken as a value that is part of transaction text which corresponds with the current field. Otherwise value is searched between list of decoded values from ENUM_VALUES field (it is the list which consists of elements of the following view separated by : Field value=Decoded value



**Type of
transaction
field**

Description

5	Date in format <YYYYMMDD>. If the number is not in this format then it is supposed that the field is not defined
6	Real number written in view of string with scale 6

8. Collaborations between XML file elements and QUIK DB fields

XML file element

Field in database table

Trans.TransNum	TRANS.TRANS_ID
Trans.UID	TRANS.USER_ID
Trans.TransID	TRANS.TRANS_TYPE
Trans.SessionID	TRANS.SESSION_ID
Trans.TradeDate	SESSIONS.TRADEDATE
Trans.Status	TRANS.TRANS_STATUS
Trans.QuikTime	TRANS.SERVER_TIME
Trans.ReplyTime	TRANS.REPLY_TIME
Trans.OrderNum	TRANS.TS_NUM
Trans.ClientCode	TRANS.CLIENT_CODE
Trans.ParentTransID	TRANS.PARENT_TRANS_ID
UserInfo.Name1	PERSON.PERSON_FIRST_NAME
UserInfo.Name2	PERSON.PERSON_MIDDLE_NAME
UserInfo.Name3	PERSON.PERSON_LAST_NAME
UserInfo.OrgCode	ORGANIZATION.ORG_CODE
UserInfo.OrgName	ORGANIZATION.ORG_NAME
Data	TRANS.TRANS_DATA
PureData	TRANS.PURE_TRANS_DATA
Reply	TRANS.TRANS_REPLY_DATA
Sign	Text of signature



XML file element	Field in database table
TransData.ClassID	TRANS_DESC.CLASS_ID
TransData.ClassCode	SECURITY_CLASS.SECURITY_CLASS_CODE
TransData.Description	TRANS_DESC.TRANS_DESCRIPTION
TransData.FieldsNum	TRANS_DESC.FIELDS_NUM
TransData.Flags	TRANS_DESC.FLAGS
TransData.InfoType	TRANS_DESC.TRANS_INFO
TransData.InfoExType	TRANS_DESC.TRANS_INFO_EX
TransData.ClassType	SECURITY_CLASS.TYPE
TransData.ClassSubType	SECURITY_CLASS. SUBTYPE
TransData.ClassBehavior	SECURITY_CLASS. BEHAVIOR
TransData.ClassSpecialBehavior	SECURITY_CLASS. SPECIAL_BEHAVIOR
Field.Number	TRANS_FIELD_DESC.FIELD_NO
Field.Name	TRANS_FIELD_DESC.FIELD_NAME
Field.Description	TRANS_FIELD_DESC.FIELD_DESC
Field.Size	TRANS_FIELD_DESC.FIELD_SIZE
Field.Type	TRANS_FIELD_DESC.FIELD_TYPE
Field.Flags	TRANS_FIELD_DESC.FLAGS
Field.DefValue	TRANS_FIELD_DESC.DEFAULT_VALUE
Field.EnumCount	TRANS_FIELD_DESC.ENUM_COUNT
Field.EnumValues	TRANS_FIELD_DESC.ENUM_VALUES
Field.Value	Value of transaction field
Field.PreparedValue	Transformed value of transaction field

9. Modification of created XML file

To transform the XML file which was created during qt2xml program work to handler format, use the attached scheme of **trans.xsl** file. To transform the XML file, add the following string to the created file after the first string:

```
<?xml-stylesheet type="text/xsl" href="trans.xsl"?>
```



As a result, the XML file will be transformed according to the scheme from trans.xsl file. The modifications will be applied when the file is opened again.

Name of the file scheme of which is used for displaying XML-file can be specified in parameter **UseXSL of [General] section. When forming the file the row with value specified in parameter must be put after the first row:**

```
<?xml-stylesheet type="text/xsl" href="<value of parameter UseXSL>"?>
```

10. Encrypting passwords

Passwords specified in the Program settings can be encrypted. The password must not begin with "\$" and "#" symbols. If you use a password that begins with any of the above symbols, change the password.

To enable the password encryption, add symbol \$ before text of the password in qt2xml.ini file.

For example:

```
Password=$secret
```

When attempting to use password, the Program will encrypt the password and save it back to the configuration file qt2xml.ini in the encrypted form. The encrypted password is marked by symbol # at the beginning of line.

When transferring Program to another computer or reinstalling the operation system, the encrypted passwords get unavailable for unencrypting. In this case the appropriate passwords must be set again.

