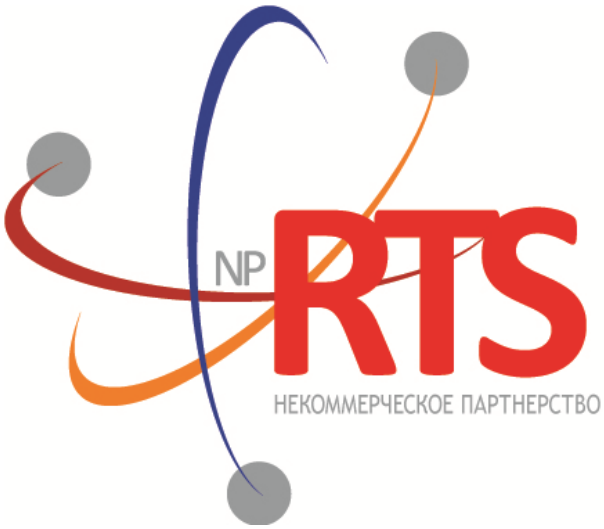

ADVANCED
TRADING & CLEARING
PLATFORM

Native Protocol Market Data Service

System version 1.7

Interface version 36

Document version 1.17.2

04 February 2019

Revision history

Version 1.17.2 February 1, 2019

Added value 4 (MemberTariff) of the fee_schema field in the Instrument message.

Version 1.17.1 December 14, 2018

1. Document structure was changed.
2. Terminology related to topic data transmission was changed.
3. Names of messages of OrderBook, Trades, CurrentPriceOfMarket, BestPrices and Commons topics are specified.
4. Description of key fields in topic messages was added.
5. The Heartbeat (msgid=15236) message was renamed to MdHeartbeat.
6. The topic_header component was renamed to header.
7. The addresses component was renamed to Report_Address.
8. The CommonEntry component was renamed to CommonsUpdateEntry.
9. The PriceLevel component was renamed to sub_aggr.
10. The BestPrice component was renamed to sub_best.

Version 1.17.0 November 3, 2017

1. The BorrowingStatus message has been added to the Instruments topic.
2. The msgid field value changed for the TradeModes message.
3. The over_the_counter field added to the TradeModes message.
4. The msgid field value changed for the Instrument message.
5. The borrowing_status field added to the Instrument message.
6. The trading_status field of the TradingInstrumentStatus message renamed to status.
7. The list of parameters in the Commons topic, unavailable for over-the-counter instruments, has been added.
8. Terminology changes.
9. Error codes added.

Version 1.16.0 November 30, 2016

1. New field markets added to the Period component.
2. The msgid value changed in the Instrument message.

Version 1.15.0 23 March 2016

The Market message is added in the Instruments topic.

Version 10.14.0 9 March 2016

1. The message broadcast in the Commons snapshot is changed.
2. The Commons topic of additional snapshot is removed.
3. New values type=73 and 75 added in the Commons message.

Table of Contents

List of Tables

1. Service overview

1.1. Data topics

The gateway currently provides the following topics:

1. OrderBook is a list of buy and sell orders for a specific instrument, grouped by price level.The number of levels is 50.
2. Trades is a list of public trades matched at the liquidity pools during the current trading day.
3. CurrentPriceOfMarket is a current market price, changing by the trade and the best order.
4. BestPrices is the top of an instrument order book—the highest bid and the lowest ask.
5. Commons is a list of statistic data of the accessed liquidity pools.
6. Instruments is instrument reference information.

Connection parameters are listed in the *Network Connectivity* document.

Messages of each topic have continuos numbering in the `topic_seq` field. The numbering of messages sent to client may be discontinuous as client receives data in accordance with login access rights.

1.2. Broadcast modes

Topics can broadcast data in two modes — **snapshot** and/or **updates**.

A snapshot is aggregation of all current data, e.g. a whole order book, transmitted at a specified frequency.

Updates are separate messages generated and transmitted to the client when an event occurs.

During a period of inactivity in an update feed the system sends a `MdHeartbeat` to acknowledge connection. If messages are not transmitted for a longer period, there is either a transmission delay or absence of connection.

1.3. Broadcast channels

Each market data topic is broadcast through two identical UDP channels — *A* and *B*. Both channels simultaneously transfer messages with the same numbers. The channel duplication provides more transmission fidelity and lowers the probability of package loss. The client is strongly recommended to process both channels. For example, if a client receives *n*+1 message after *n*–1 message in *A* channel, then *n* message will be probably found in *B* channel. If a package is lost in the both channels, a client should either wait the next snapshot or request the message via recovery gateway.

1.4. Algorithm of receiving and processing topic data

If you want to connect to a topic with snapshots and updates, it is recommended to connect in both modes. First, you should receive a complete snapshot, then start recording incoming updates. You are recommended to record messages from both UDP-channels (A and B) and sort them by number. If an update has been lost in one of the channels, it can be requested in recovery gateway (messages, missing from the snapshots, cannot be recovered). . There is delay between appearing of the same messages in market data gateway and messages in recovery gateway.If a significant number of messages is lost, it is recommended to request the snapshot instead of attempting to recover lost updates.

When snapshot is complete you should record the updates. Updates can replace or replenish earlier data, depending on the topic. For topics with replacement there are identifiers of updated data - `keys`. The `keys` are fields values of topic messages and are indicated in header of tables in section 3.

Table 1. Features of snapshot and updates

Topic	Update		Snapshot
	Replenishment	Replacement	
Trades	✓		Messages history since the start of the trading day
OrderBook CurrentPriceOfMarket BestPrices Commons	✓	✓	An aggregation of all current data
Instruments		✓	

1.4.1. Example for `OrderBook` topic

Updates from `OrderBook` topic **replace** earlier data.

1. Connect to the updates mode of the required topic and save all incoming `AggrMsgOnline` messages.
2. Connect to the snapshot mode of the topic and wait for `SnapshotStarted`.
3. Save all incoming `AggrMsgSnapshot` messages, until `SnapshotFinished` is received.
   - If some `AggrMsgSnapshot` messages have been skipped, or `update_seq` values are different for `SnapshotStarted` and `SnapshotFinished` messages, then repeat actions **2** and **3**.
   - If there is no saved `AggrMsgOnline` message with the number equal to `update_seq+1` , then repeat actions **2** and **3**.
4. Compare keys values of `AggrMsgSnapshot` and each `AggrMsgOnline` with number `seq>update_seq` (the keys of `OrderBook` topic are `instrument` and `source_id`):
   - If the values are equal, you should replace the snapshot message with `AggrMsgOnline`.
   - If the values aren't equal, you should replenish the snapshot with `AggrMsgOnline`.

1.4.2. Example for `Trades` topic

For example the client connect to `Trades` topic during the trade day (sequential numbers of messages are reset every night). Number of the last `Trade` message is 105.

Updates from `Trades` **replenish** earlier data.

1. Connect to `Trades` topic and wait for first trade message. If `Trade` message is not available withing 5 seconds after the connection, then the `MdHeartbeat` message is received.
2. Obtain sequential number `seq` of the first received message (for example, `MdHeartbeat` message with `seq=305`). You can determine, that messages from `seq=106` to `seq=304`, were not received.
3. To recover messages you schould connect to the gateway and send `TopicRequest` with the `topic=Trades`, `topic_seq=106`, and `topic_seqend=304`.
4. The `TopicRequest` will result in the following message sequence:
   - TopicReport (seq=0, Start);
   - Trade (seq=1, topic_seq=150);
   - Trade (seq=2, topic_seq=170);
   - Trade (seq=3, topic_seq=200);
   - Trade (seq=4, topic_seq=303);
   - TopicReport (seq=0, End).

The broadcast of recovered data is surrounded by the `TopicReport` messages. The `Trade` messages have gaps between `topic_seq` values, because the `Heartbeat` messages were received between `Trade` messages.

2. Protocol overview

2.1. Data types

The trading system uses little-endian byte order (same as in x86 processor); the client shall use same.

`asciiN` is an alphanumeric string of *N*-byte length; the unused part should be filled with zero bytes.

`charN+1` is a UFT-8 encoded string of *N+1*-byte length. The last byte is the end of line character and so the available length is *N*; the unused part should be filled with zero bytes.

`dec2` is an eight-byte integer representing a fraction multiplied by 10^2.

`dec8` is an eight-byte integer representing a fraction multiplied by 10^8.

`decn` is a nine-byte sequence; the first eight bytes are an integer representing a fraction multiplied by 10^n and the last byte is *n*. Its value should be within the range from 0 to 8.

`intN` is an *N*-byte integer.

`time4` is a four-byte integer representing the Unix time in seconds, i.e. the number of seconds since 1 January 1970.

`time8n` is an eight-byte integer representing the Unix time in nanoseconds, i.e. the number of nanoseconds since 1 January 1970.

`time8m` is an eight-byte integer representing the Unix time in milliseconds, i.e. the number of milliseconds since 1 January 1970. If a field of this datatype conveys a date, the value part representing hours, minutes, seconds and milliseconds should be neglected, i.e. that is to use an integer value (rounded down) of division by 86 400 000.

2.2. Message format

A native protocol message is a sequence of field values in a strict order. Each message starts with the `frame` header; this three-field component includes message size, message type, and sequence number. The message size is the length of the whole message, except for the frame header, in bytes. The size is constant for all message types which don not include any repeating component or field.

A message is transmitted in a network packet as a sequence of bytes.

2.3. Common components of messages

Table 2. Format of component `frame`: length 12 bytes

Field	Datatype	Description
size	int2	Message length in bytes, excluding the `frame` header
msgid	int2	Message type
seq	int8	Application message sequence number

Common components for topics

Table 3. Format of component `instrument`: length 6 bytes

Field	Datatype	Description
market_id	int2	Liquidity pool ID (please refer to section 2.7)
instrument_id	int4	Trading instrument ID

Table 4. Format of component `md_header`: length 10 bytes

Field	Datatype	Description
system_time	time8n	Timestamp of message generation
source_id	int2	Message source (for values please refer to section 2.6)

Common components for recovery gateway

Table 5. Format of component `user_header`: length 20 bytes

Field	Datatype	Description
clorder_id	ascii20	Client order ID

Table 6. Format of component `gate_header`: length 46 bytes

Field	Datatype	Description
system_time	time8n	Client request processing time
source_id	int2	Message source (for values please refer to section 2.6)
clorder_id	ascii20	Client order ID
user_id	ascii16	Login, client gateway ID

Table 7. Format of component `header`: length 22 bytes

Field	Datatype	Description
topic_id	int4	Numerical ID of topic
topic_seq	int8	Message sequence number in topic
system_time	time8n	Message generation time
source_id	int2	Message source (for values please refer to section 2.6)

2.4. Recovered message format

The format of a recovered message is identical to that of a broadcast message, except for the header — the `header` stands for the `md_header`. Therefore, offsets of all following fileds are increased by 12 bytes. This is due to the recovered message are transmitted via TCP, not UDP.

Broadcast message format

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Market data header
10	[instrument]	[instrument]	Trading instrument

Recovered message format

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[header]	[header]	Header
22	[instrument]	[instrument]	Trading instrument

Figure 1. Message format alternation

2.5. Repetitive components and fields

Several message types contain one or more repeating components or fields which may have an arbitrary number of entries. One message may include multiple repetitive components and fields. All same-type repetitive components has a constant length.

A repeating component or field is always preceded by the two fields — `offset` and `count`. The `count` field specifies the number of entries. The `offset` field indicates an offset in bytes of first entry from the beginning of this very field; its value is no less than 4.

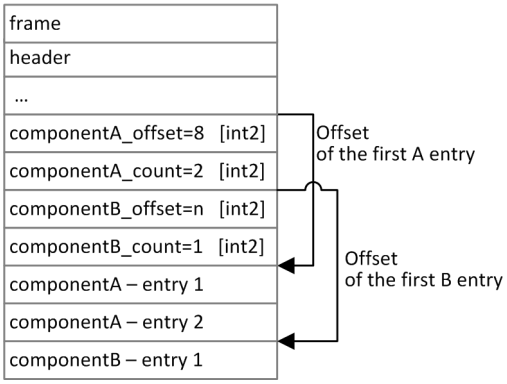

Figure 2. Template of a message with two repeating components

A repeating component may include another repeating component or field. In this case each entry refers to its own set of the embedded entries.

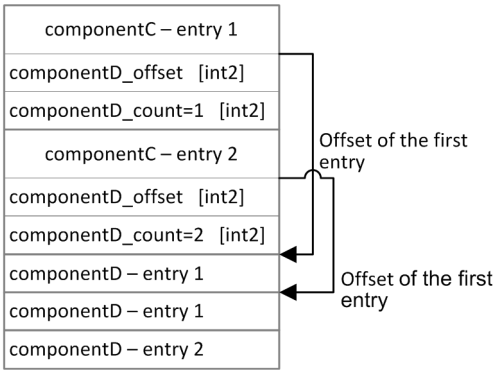

Figure 3. Template of embedded components

2.6. `Source_id` values

Field `source_id` is in the headers `gate_header`, `md_header` and `header`; the field specifies the module transmitting message to gateway for sending it to client.

Table 8. `Source_id` values to be returned to client

Range	Description
100–199	Trading system gateway
200–249	Clearing House risk parameter verification modules
250–259	Matching modules
300–499	Modules of generation and calculation of market data
500–549	Routing modules
1000–1099	Liquidity pools identifiers

2.7. Liquidity pool identifiers

Liquidity pools' identifiers may be in fields `markets`, `market_id` and `source_id` .

0 (DEFAULT) — liquidity pool is defined by the trading system.

1001 (TRADSYS) — all available liquidity pools.

1000 — liquidity pool of Saint-Petersburg Exchange.

1010 — liquidity pool of Moscow Exchange.

1015 — execution at United States liquidity pools.

1016 — market data from United States liquidity pools.

1030 — liquidity pool of NYSE.

1031 — liquidity pool of ARCA.

1032 — liquidity pool of NASDAQ.

1033 — liquidity pool of BATS.

3. Market data messages

3.1. Snapshot start and finish

For all channels, a snapshot is preceded by a `SnaphotStarted` and followed by a `SnapshotFinished`. The both messages contains the `updates_seq` field that conveys the sequence number of the last update message involved in the snapshot. Therefore, the update messages to be applied to the snapshot have a `seq` grater than the `updates_seq`.

Table 9. Format of message `SnapshotStarted`: msgid=12345, size=18

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	update_seq	int8	Sequence number of the last update included in the snapshot

Table 10. Format of message `SnapshotFinished`: msgid=12312, size=18

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	update_seq	int8	Sequence number of the last update included in the snapshot

3.2. OrderBook topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The OrderBook snapshot conveys 50 or less price levels; the updates concern 50 disclosed price levels only.

An OrderBook message concerns the order book of an instrument which is specified in the `instrument` component.

The updates are transmitted via the `AggrMsgOnline` messages, the snapshots are transmitted via the `AggrMsgSnapshot` messages.

The final part of an OrderBook message is the `sub_aggr` repeating component with the number of entries specified in the `PriceLevel_count` field (for more information on processing of repeating component please refer to section 2.5). A component entry includes price level, orders direction, add/update indicator, total disclosed amount of orders at the price level, and latest update timestamp.

The value of the `flag` field indicates whether the price level is added or updated; and a price level removal will be described as amount update to zero. In a snapshot, price levels are defined as new.

Table 11. Format of message `AggrMsgOnline`: msgid=1111, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	aggr_offset	int2	Offset of the first `aggr` entry from the beggining of this field

Offset	Field	Datatype	Description
18	aggr_count	int2	Number of the `aggr` group entries
	> aggr	[sub_aggr]	List of price levels

Table 12. Format of message `AggrMsgSnapshot`: msgid=1112, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	aggr_offset	int2	Offset of the first `aggr` entry from the beggining of this field
18	aggr_count	int2	Number of the `aggr` group entries
	> aggr	[sub_aggr]	List of price levels

Table 13. Format of component `sub_aggr`: length 22 bytes

Field	Datatype	Description
price	dec8	Price
type	int1	Sides of orders. Values: • `1` (BUY_DIR): buy; • `2` (SELL_DIR): sell; • `3` (LAST_DEAL): last trade
flag	int1	Flag of new entry. Values: • `0x0` (UPDATE): updating; • `0x1` (NEW): adding
amount	int4	Total volume at the price level
time	time8n	Time of recent price level change

The gateway sends the `EmptyBook` message to the OrderBook topic for an order book clearing after a trading system restart.

Table 14. Format of message `EmptyBook`: msgid=15300, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument

3.3. Trades topic

*Snapshot is an entire message history since the start of the trading day. Updates **replenish** earlier data.*

Upon a trade execution, the trading system generates a `Trade` message containing trade parameters with the liquidity pool of execution in the `market` field of the `instrument` component, with a unique trade identifier `trade_id`, with the trade volume `amount`, with the trade price `price`, with the transaction timestamp `trade_time`, and with taker's order side `dir`.

Table 15. Format of message `Trade`: msgid=15210, size=46, keys=instrument, trade_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	trade_id	int8	Trade ID assigned by liquidity pool
24	amount	int4	Trade volume
28	price	dec8	Trade price
36	trade_time	time8n	Trade time
44	trade_type	int1	Trade type. Value: `1` (REGULAR): regular trade
45	dir	int1	Initiator"s order side. Values: • `1` (Buy): buy; • `2` (Sell): sell

3.4. CurrentPriceOfMarket topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The `Trade` message in CurrentPriceOfMarket topic is created, when the current market price changes. The message includes: new `price`, time of price change `trade_time` and deal side `dir`.

The current price is continuously calculated, based on deal prices and hard quotes according to the following rules:

1. If a deal is made, the current price becomes equal to the deal price.
2. If an anonymous buying order appears in the order book, and its price is higher, than the current market price, the current market price becomes equal to the buying order price.
3. If an anonymous selling order appears in the order book, and its price is lower, than the current market price, the current market price becomes equal to the selling order price.

Table 16. Format of message `Trade`: msgid=15210, size=58, keys=instrument, trade_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification

Offset	Field	Datatype	Description
16	trade_id	int8	Trade identifier assigned by liquidity pool
24	amount	int4	Trade volume
28	price	dec8	Trade price
36	trade_time	time8n	Transaction timestamp
44	trade_type	int1	Trade type: • `1` (REGULAR)
45	dir	int1	Taker's order side: • `1` (BUY) • `2` (SELL)

3.5. BestPrices topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The BestPrices snapshot conveys the best offer, the best bid, and the latest trade. One message relates a trading instrument; the liquidity pool and the instrument are specified in the `instrument` component.

The updates are transmitted via the `PricesOnline` messages, the snapshots are transmitted via the `PricesSnapshot` messages.

The final part of a BestPrice message is the `sub_best` repeating component with the number of entries specified in the `sub_prices_count` field (for more information on processing of repeating component please refer to section 2.5). The component entry includes price level, orders direction, add/update indicator, total disclosed amount of orders at the price level, and latest update timestamp.

Table 17. Format of message `PricesOnline`: msgid=7651, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	sub_prices_offset	int2	Offset of the first `sub_prices` entry from the beggining of this field
18	sub_prices_count	int2	Number of the `sub_prices` group entries
	> sub_prices	[sub_best]	List of the best price levels

Table 18. Format of message `PricesSnapshot`: msgid=7653, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
16	sub_prices_offset	int2	Offset of the first `sub_prices` entry from the beg-gining of this field
18	sub_prices_count	int2	Number of the `sub_prices` group entries
	> sub_prices	[sub_best]	List of the best price levels

Table 19. Format of component `sub_best`: length 22 bytes

Field	Datatype	Description
price	dec8	Price
type	int1	Entry type. Values: • `1` (BEST_BUY): best price to buy; • `2` (BEST_SELL): best price to sell; • `3` (LAST_DEAL)
pad0	ascii1	Field reserved for future. Zero byte value
amount	int4	Total volume of orders at level or volume of trade
time	time8n	Time of recent price level change or trade time

3.6. Commons topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The Commons topic transmits various market parameters, see the list below. A Commons message concerns a single trading instrument; the liquidity pool and the instrument are specified in the `instrument` component.

The updates are transmitted via the `CommonsUpdateOnline` messages, the snapshots are transmitted via the `CommonsUpdateSnapshot` messages.

The messages contain the `CommonsUpdateEntry` repeating component and each entry describes a parameter. The datatype of the `value` field depends on the `type`. The number of entries is specified in the `entry_count` field (for more information on processing of repeating component please refer to section 2.5).

Snapshots are transmitted in succession. An update is generated on data change.

Table 20. Snapshot and update parameters correspondence

Parameter type	Update `type` field	Update `value` field
Latest trade price	3	dec8
Opening price	4	dec8
Closing price	5	dec8
Highest price	7	dec8
Lowest price	8	dec8
Closing auction price of previous day	73	dec8
Halt price	74	dec8

Parameter type	Update `type` field	Update `value` field
Timestamp of lowest current price	75	time8n
Indicative quote	76	dec8
Turnover of trades at closing auction price, in instrument units	79	int8
Turnover for calculation of previous day's market price 3	80	dec2
Turnover for calculation of current day's market price 3	81	dec2
Turnover for calculation of previous day's market price 2	82	dec2
Turnover for calculation of current day's market price 2	83	dec2
Timestamp of current price calculation	84	time8n
Change in current price against official closing price of previous day	85	dec8
Lowest current price	86	dec8
Latest trade price involved in current price	87	dec8
Volume disbalance at closing auction	88	int8
Market price 3 of previous day	89	dec8
Market price 3 of current day	90	dec8
Market price 2 of previous day	91	dec8
Market price 2 of current day	92	dec8
Last trade price of main session in previous day	93	dec8
Last trade price of main session in current day	94	dec8
Turnover of latest trade in price currency	95	dec2
Official closing price of previous day	96	dec8
Official current price	97	dec8
Volume weighted average price of main session in previous day	98	dec8
Volume weighted average price of main session in current day	99	dec8
Current price	100	dec8
Settlement price of latest main clearing	101	dec8
Settlement price of latest intermediate clearing	102	dec8
Number of buy orders	103	int8
Number of sell orders	104	int8
Volume of buy orders	105	int8
Volume of sell orders	106	int8

Parameter type	Update `type` field	Update `value` field
Number of anonymous trades	107	int8
Turnover of anonymous trades, in instrument lots	108	int8
Turnover of anonymous trades, in instrument units	109	int8
Currency turnover of anonymous trades	110	dec2
Total number of trades	111	int8
Total turnover, in instrument lots	112	int8
Total asset turnover	113	int8
Total currency turnover	114	dec2
Closing auction price	115	dec8
Closing auction volume	116	int8
Volume weighted average price	117	dec8
Highest bid price	118	dec8
Lowest ask price	119	dec8
Volume of latest trade	120	int8
Timestamp of latest trade	121	time8n
Price of last trade over previous trading period	122	dec8

Table 21. Parameters unavailable in over-the-counter trades

Parameter type	Update `type` field	Update `value` field
Closing price	5	dec8
Halt price	74	dec8
Timestamp of lowest current price	75	time8n
Turnover for calculation of previous day's market price 3	80	dec2
Turnover for calculation of current day's market price 3	81	dec2
Turnover for calculation of previous day's market price 2	82	dec2
Turnover for calculation of current day's market price 2	83	dec2
Timestamp of current price calculation	84	time8n
Change in current price against official closing price of previous day	85	dec8
Lowest current price	86	dec8
Latest trade price involved in current price	87	dec8
Market price 3 of previous day	89	dec8
Market price 3 of current day	90	dec8

Parameter type	Update `type` field	Update `value` field
Market price 2 of previous day	91	dec8
Market price 2 of current day	92	dec8
Last trade price of main session in current day	94	dec8
Official closing price of previous day	96	dec8
Official current price	97	dec8
Volume weighted average price of main session in previous day	98	dec8
Volume weighted average price of main session in current day	99	dec8

Table 22. Format of message `CommonsUpdateOnline`: msgid=1113, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	entry_offset	int2	Offset of the first `entry` entry from the beggining of this field
18	entry_count	int2	Number of the `entry` group entries
	> entry	[CommonsUpdateEntry]	List of commons

Table 23. Format of message `CommonsUpdateSnapshot`: msgid=1115, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	entry_offset	int2	Offset of the first `entry` entry from the beggining of this field
18	entry_count	int2	Number of the `entry` group entries
	> entry	[CommonsUpdateEntry]	List of commons

Table 24. Format of component `CommonsUpdateEntry`: length 10 bytes

Field	Datatype	Description
type	int1	Entry type.
flags	int1	Flag of value. Values: • `0x0` (NORMAL): valid; • `0x1` (DELETE): deleted

Field	Datatype	Description
value	int8	Entry value (valid when flags=0)

3.7. Instruments topic

*Snapshot is aggregation of all current data. Updates **replace** earlier data.*

The Instrument topic broadcasts reference data on trading instruments:

- `Currency` balance instrument,
- `Issue` balance instrument,
- `Spot` balance instrument,
- `Futures` balance instrument,
- `Bond` balance instrument.
- `TradeModes`,
- `Markets`
- trading `Instrument`.

The Instrument snapshot and updates transmits the same messages. The update of the Instruments topic broadcasts the `TradingInstrumentsStatus` message on instrument status change and the `TradingInstrumentLimits` on price limit change. The `BorrowingStatus` message is sent, when short selling availability of an instrument has changed.

The Instruments topic cannot be recovered over TCP.

Table 25. Format of message `Currency`: msgid=931, size=266, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Currency code
47	desc	char64+1	Full name of currency in English
112	desc_ru	char128+1	Full name of currency in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of asset
258	cfi_code	char6+1	CFI code
265	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 26. Format of message `Issue`: msgid=932, size=474, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header

Offset	Field	Datatype	Description
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Instrument ticker
47	desc	char64+1	Full name of stock in English
112	desc_ru	char128+1	Full name of stock in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of lot
258	isin	char32+1	ISIN
291	cfi_code	char6+1	CFI code
298	reg_num	char32+1	Registration number
331	issuer_name	char64+1	Name of issuer or management company (for stakes)
396	issuer_country	char8+1	Issuer country
405	face_value	dec8	Face value
413	face_value_currency	char8+1	Face value currency
422	total_amount	decn	Total amount of issue
431	security_type	int1	Security type. Values: • `1` (OrdinaryShare): ordinary share; • `2` (PreferredShare): preferred share; • `5` (ETF): security of foreign exchange traded fund; • `6` (RDR): Russian depositary receipt; • `7` (ADR): American depositary receipt; • `8` (GDR): global depositary receipt; • `9` (IntervalMutualFund): share of mutual fund
432	issue_date	time8m	Issue or registration date
440	quotation_list	char32+1	Quotation list
473	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 27. Format of message `Spot`: msgid=933, size=281, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Spot code

Offset	Field	Datatype	Description
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	lot	int8	Lot volume in balance instrument units (instrument ID specified in `underlying_id`)
258	date_exec	time8m	Execution date
266	shift	int2	Shift of execution date from today
268	underlying_id	int4	Underlying instrument ID
272	accrued_interest	dec8	Accrued interest as of the delivery date
280	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 28. Format of message `Futures`: msgid=934, size=280, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Futures code
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	lot	int8	Lot volume in balance instrument units (instrument ID specified in `underlying_id`)
258	date_exec	time8m	Execution date
266	date_expire	time8m	Expiration date
274	underlying_id	int4	Underlying instrument ID
278	exec_type	int1	Futures type. Values: • `0` (FuturesThroughSpot): futures through spot; • `1` (FuturesCashSettlement): cash-settled futures
279	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 29. Format of message `Bond`: msgid=935, dynamic length, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Bond code
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of lot
258	isin	char32+1	ISIN
291	cfi_code	char6+1	CFI code
298	date_maturity	time8m	Maturity date
306	coupon_payment_offset	int2	Offset of the first `coupon_payment` entry from the beggining of this field
308	coupon_payment_count	int2	Number of the `coupon_payment` group entries
310	reg_num	char32+1	Registration number of bond issue
343	issuer_name	char64+1	Name of issuer or management company (for stakes)
408	issuer_country	char8+1	Issuer country
417	face_value	dec8	Face value
425	face_value_currency	char8+1	Face value currency
434	issue_amount	decn	Total amount of issue
443	security_type	int1	Security type. Values: • `1` (GovernmentBond): government bond; • `2` (MunicipalBond): municipal bond; • `3` (CentralBankBond): Central bank bond; • `4` (CorporateBond): corporate bond; • `5` (FinancialInstitutionBond): financial institution bond
444	issue_date	time8m	Date of issue
452	quotation_list	char32+1	Quotation list
485	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test
	> coupon_payment	[coupon_payment]	Schedule of coupon payments

Table 30. Format of message `TradeModes`: msgid=942, size=210, keys=trade_mode_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	trade_mode_id	int2	Trade mode ID
12	name	char64+1	Name of trade mode in English
77	name_ru	char128+1	Name of trade mode in Russian
206	is_address	int1	Negotiated trading flag in trade mode. Values: • `0` (No): non-negotiated; • `1` (Yes): negotiated
207	is_multileg	int1	Multi-leg trade indicator. Values: • `0` (No): sigle-leg; • `1` (Yes): multi-leg
208	is_ext_close	int1	Closing auction indicator. Values: • `0` (No): not traded at closing auction; • `1` (Yes): traded at closing auction
209	over_the_counter	int1	Over-the-counter trade mode indicator. Values: • `0` (No): not present; • `1` (Yes): present

Table 31. Format of message `Market`: msgid=936, size=208, keys=market_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	market_id	int4	Market_id
14	desc	char64+1	Full name in English
79	desc_ru	char128+1	Full name in Russian

Table 32. Format of message `Instrument`: msgid=973, dynamic length, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	symbol	char32+1	Symbolic instrument ID
47	desc	char64+1	Full instrument name in English

Offset	Field	Datatype	Description
112	desc_ru	char128+1	Full instrument name in Russian
241	status	[instrument_status]	Current status of trading instrument
245	type	char3+1	Trading instrument type: • f: futures; • t: T+N; • o: option; • r: repo; • pr: related trades; • sw: swap; • c: calendar spread; • sf: spot-futures spread; • dvp: delivery versus payment
249	auction_dir	int1	Type of auction. Values: • `0` (Direct): direct auction; • `1` (Inverse): inverse auction
250	price_increment	dec8	Price increment
258	step_price	dec8	Step price
266	legs_count	int2	Number of legs
268	trade_mode_id	int2	Trading mode ID
270	scalping_type	int2	Scalping type. Values: • `0` (NoScalping): no scalping; • `1` (Custom): custom scalping; • `2` (InverseScalping): inverse scalping
272	fee_schema	int1	Fee scheme. Values: • `1` (MakerTakerSpot): maker-taker for spot; • `2` (MakerTakerFutures): maker-taker for futures; • `3` (REPO): repo; • `4` (MemberTariff): maker-taker for spot by members
273	fee_rate_offset	int2	Offset of the first `fee_rate` entry from the beggining of this field
275	fee_rate_count	int2	Number of the `fee_rate` group entries
277	curr_price	char16+1	Currency of the instrument price
294	periods_offset	int2	Offset of the first `periods` entry from the beggining of this field
296	periods_count	int2	Number of the `periods` group entries

Offset	Field	Datatype	Description
298	exchange_instrument_offset	int2	Offset of the first `exchange_instrument` entry from the beggining of this field
300	exchange_instrument_count	int2	Number of the `exchange_instrument` group entries
302	limit_up	dec8	Price limit up
310	limit_down	dec8	Price limit down
318	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test
319	te_id	int2	Trading engine ID
321	be_mode	int1	Best execution mode. Values: • `0` (External): external trades; • `1` (Internal): internal trades at external prices
322	borrowing_status	int1	Short selling availability for the instrument. Values: • `1` (HARD_TO_BORROW): short selling unavailable; • `2` (EASY_TO_BORROW): short selling available
	> fee_rate	dec8	Fee rate
	> periods	[Period]	Component of trading periods (such as trading session) for instrument
	> exchange_instrument	[ExchangeInstrument]	Component specifying trading instruments at liquidity pools

In this version of the trading system, the `fee_rate` group has five entries. The group has the following sequence of entries:

1. Minimum fee rate, in instrument currency.
2. Fee rate for pre-delivery trades, in instrument currency.
3. Taker fee rate depending on fee scheme: portion of trade volume in price currency for shares; amount of price currency per contract for derivatives; portion of the first leg value multiplied by repo duration for repo.
4. Maker fee rate depending on fee scheme: portion of trade volume in price currency for shares; amount of price currency per contract for derivatives; portion of the first leg value multiplied by repo duration for repo.
5. Accuracy.

Values of third and fourth records are based on the mechanism of fee calculation specified in the fee_schema field .

Table 33. Format of message `TradingInstrumentStatus`: msgid=2031, size=84, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
16	trading_status	int1	Status of trading instrument. Values: • 2 (HALT): trading is halted; • 17 (TRADING): trading in progress; • 18 (NO_TRADING): no trading; • 102 (CLOSE): trading during closing auction; • 103 (CLOSE_PERIOD): trading during close period; • 107 (DISCRETE_AUCTION): trading during discrete auction; • 118 (OPEN): trading during opening auction; • 120 (FIXED_PRICE_AUCTION): trading at closing auction price
17	reserved	char2+1	Reserved field. To be filled with null byte
20	comment	char63+1	Comments

Table 34. Format of message `TradingInstrumentLimits`: msgid=2032, size=30, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	limit_up	dec8	Price limit up
22	limit_down	dec8	Price limit down

Table 35. Format of message `BorrowingStatus`: msgid=2033, size=15, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	borrowing_status	int1	Short selling availability for the instrument. Values: • 1 (HARD_TO_BORROW): Short selling unavailable; • 2 (EASY_TO_BORROW): Short selling available

Table 36. Format of component `coupon_payment`: length 16 bytes

Field	Datatype	Description
date	time8m	Date of payment
value	dec8	Amount of payment

Table 37. Format of component `Period`: length 30 bytes

Field	Datatype	Description
start	time8m	Start timestamp
finish	time8m	End timestamp
mode	int2	Type of auction. Values: • `0` (ProRata): pro rata two-way anonymous auction; • `1` (Parity): parity two-way anonymous auction; • `2` (TimePriority): time priority anonymous auction; • `3` (Address): negotiated trading; • `4` (OpenAuction): opening auction; • `5` (CloseAuction): closing auction; • `6` (NoTrade): no trading; • `7` (ExtClose): closing auction at liquidity pool
currency_id	int4	Currency ID of traded instrument
underlying_offset	int2	Offset of the first `underlying` entry from the beggining of this field
underlying_count	int2	Number of the `underlying` group entries
markets_offset	int2	Offset of the first `markets` entry from the beggining of this field
markets_count	int2	Number of the `markets` group entries
> underlying	[Underlying]	Component for trading instrument lot volume specification within a period of time
> markets	int2	List of available liquidity pools (please refer to section 2.7)

Table 38. Format of component `ExchangeInstrument`: length 61 bytes

Field	Datatype	Description
instrument	[instrument]	Component specifying trading instrument
code_group	char16+1	Market section
code	char16+1	Instrument ticker
code_extra	char16+1	Instrument code
status	[instrument_status]	Current status of trading instrument

Table 39. Format of component `instrument_status`: length 4 bytes

Field	Datatype	Description
trading_status	int1	Current status of trading instrument. Values: • `2` (HALT): trading is halted; • `17` (TRADING): trading in progress; • `18` (NO_TRADING): no trading; • `102` (CLOSE): trading during closing auction; • `103` (CLOSE_PERIOD): trading during close period; • `107` (DISCRETE_AUCTION): trading during discrete auction; • `118` (OPEN): trading during opening auction; • `120` (FIXED_PRICE_AUCTION): trading at closing auction price
suspend_status	int1	Reserved field. To be filled with null byte
routing_status	int1	Reserved field. To be filled with null byte
reason	int1	Reserved field. To be filled with null byte

Table 40. Format of component `Underlying`: length 15 bytes

Field	Datatype	Description
balance_id	int4	Balance instrument ID
qty	decn	Number of balance instrument units
flags	int2	Flags field. Values: • `0x1` (CORP_DUE_BILL): additional liability in connection with corporate event; • `0x2` (CORP_CORRECTION): liability adjustment by clearing center in connection with corporate event; • `0x4` (CORP_INCOME_RETURN): transfer of income in connection with corporate event; • `0x8` (PRINCIPAL_OBLIGATION): principal liability flag

3.8. Heartbeat message

The gateway sends a `MdHeartbeat` in the topic updates if no message is transmitted for more than a second.

Table 41. Format of message `MdHeartbeat`: msgid=15236, size=14

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	reserved	int4	Reserved field. Zero value

4. Recovery gateway

The recovery gateway allows the client to request resending of update messages, if they were lost via UDP. You can request the following topics' updates via the recovery gateway: OrderBook, Trades, BestPrices, Commons and CurrentPriceOfMarket.

The whole history from the start of a trade day is available for recovery only for Trades and CurrentPriceOfMarket topics; for all other topics a client can recover only recent messages. Due to technological limitations, messages from the previous trade day can be recovered.

Client should use the discovery service to connect to the recovery gateway.

4.1. Session layer

4.1.1. Discovery service

The Discovery service provides a host address for client connections to the trading system gateway. The client should request the service for address allocation each time before connecting to the gateway. Upon receipt of response, the client should disconnect from the login server and connect to a gateway through the received address.

For the address for accessing the Discovery service please refer to *Network Connectivity*.

After establishing connection with the Discovery service, the client should send the `Hello` message. The message contains the session header `frame` (for more details refer to section 2.2). The client should specify login and password, and the IP address of the client must be authorized for the specified login (user ID).

Table 42. Format of message `Hello`: msgid=1, size=32

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login
16	password	ascii16	Password

In response to request, the server sends the `Report` message. If this message has `status`=0, the message contains repetitive component `Report_Address`; the number of component records will be specified in the field `addresses_count` (for more details on processing of repeating groups please see section 2.5). The component includes fields `type` (gateway attribute) and `address` (host address and gateway port). Gateway attributes may combine.

After the trading system responds, the gateway will expect the client's login connection to the specified address. In case of failure, the client is recommended to make two additional connection attempts with an interval of half a second. If the login is invalid or blocked, the server response will contain `status=1`.

Table 43. Format of message `Report`: msgid=2, dynamic length

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	status	int2	Request status. Values: • `0` (Success): success; • `1` (Fail): reject due to invalid login/password
2	reason	char127+1	Textual description
130	addresses_offset	int2	Offset of the first `addresses` entry from the beggining of this field
130	addresses_count	int2	Number of the `addresses` group entries

Offset	Field	Datatype	Description
	> addresses	[Report_Address]	Address list

Table 44. Format of component `Report_Address`: length 52 bytes

Field	Datatype	Description
type	int2	Gateway attributes, bit mask. Values: • `0x1` (Transaction): trading; • `0x2` (DropCopy): drop-copy; • `0x4` (Risk): risk management; • `0x8` (Dictionary): dictionaries; • `0x10` (MarketData): market data recovery; • `0x4000` (Backup): backup
ver	int1	Interface version
pad0	int1	Reserved field, filled with zero bytes
address	char47+1	Address of host and gateway port

4.1.2. Session initialization

A session is established over a network connection between the client's system and the gateway of the trading system.

Once connection is established, the client can send the `Login` message to initiate a session. The message includes the user ID and the password. The system validates the authentication parameters and answers with the `Logon` message and so the session is active. Upon receipt of a malformed `Login` message or invalid login/password, the server breaks the connection.

A login may have a single concurrent session. If the server detects a second connection attempt via the same login while a valid session is already underway, the server will respond with `Reject`.

Table 45. Format of message `Login`: msgid=8001, size=37

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login
16	password	ascii16	Password
32	reset_seq	int1	Reset sequence numbers indicator. Values: • `0` (no): sequence numbers continue; • `1` (yes): sequence numbers reset
33	heartbeat_ms	int4	`Heartbeat` frequency in milliseconds

Table 46. Format of message `Logon`: msgid=8101, size=24

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	last_seq	int8	Last application message available to client. If altered form the last received message, `ResendRequest` is to be sent

Offset	Field	Datatype	Description
8	expected_seq	int8	Next application message expected from client
16	system_id	ascii8	Deployment ID

4.1.3. Heartbeat message

The client and the gateway exchange `Heartbeat` messages to monitor the connection status. Heartbeat is sent, if no session or application message has been sent within the heartbeat interval.

When initiating a session, the client sets the heartbeat interval in the field `heartbeat_ms` of the `Login` message.

If the server detects inactivity for a period longer than the specified interval, the system will break the connection. The client is expected to do the same, if inactivity is detected on the part of the server.

Table 47. Format of message `Heartbeat`: msgid=8103, size=0

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header

4.1.4. Message numbers

All application messages have a unique number throughout the trading day. Messages by each session side (the client and the gateway) are sequentially numbered with positive integers starting with 1. This allows to request and resend messages lost in case of unexpected disconnection.

Sequence numbers are not assigned to session messages — the `seq` value is always 0.

In order to maintain sequential numbering of messages, at session initialization the gateway provides two key values in its `Logon` message — the number of the last message sent (`last_seq`) and the expected number of the following message (`expected_seq`).

If the message number differs from the expected one, the gateway terminates the connection. After disconnection, the client should reconnect by addressing the Discovery service and restore the number of messages according to the values obtained in the `Logon` message from the gateway. The gateway never initiates a change in numbering when receiving a message with the number higher than expected.

The trading system supports continuous message numbering between trading sessions, including trading days. The client should set `reset_seq=1` in message `Login` at session initialization to reset numbering.

4.1.5. Session termination

The server or the client sends `Logout` to terminate the session and expects the other party to disconnect.

Table 48. Format of message `Logout`: msgid=8002, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login, client gateway ID

4.1.6. Message rejection

If the client's message is either malformed or contains invalid values, the system rejects such message and responds with `Reject`. The `ref_msgid` field specifies message type, `ref_seq` contains the application level message number or has 0 for session message, fields `reason` and `message` contain, correspondingly, code of rejection reason and its description.

Table 49. Format of message `Reject`: msgid=8102, size=45

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	ref_seq	int8	Sequence number of rejected message
8	ref_msgid	int2	Type of rejected message
10	reason	int2	Code of rejection reason
12	message	char32+1	Rejection parameters or textual description

4.1.7. Disconnection

System disconnects when receiving message:

- with unknown value of `msgid`,
- with a `size` incorrect for the specified message type,
- with a `seq` number other than expected.

4.1.8. Data request

To request data, client should send `TopicRequest` to the trading system gateway specifying `topic` ID. The client does not have to fill the `clorder_id` field.

The client can specify the range of requested messages through `topic_seq` and `topic_seqend` fields:

- `topic_seq=n, topic_seqend=m` — request for messages from *n* to *m*.
- `topic_seq=0, topic_seqend=n` — request for messages from the lowest number available to *n*.
- `topic_seq=n, topic_seqend=0` — request for messages from *n* to the last number available.
- `topic_seq=0, topic_seqend=0` — request for all available messages.

When making an initial request for Trades topic, the client should specify 0 in `topic_seq` and `topic_seqend` fields. And in a repeating request, value of the `topic_seq` field should be one more than value of the `topic_lastseqsent` field in the last received `TopicReport`. If `TopicReport` is not recieved, value of the `topic_seq` field should be one more than that of the last message received.

When requesting for OrderBook, CurrentPriceOfMarket, BestPrices, Commons, Instruments, the client should specify 0 in `topic_seq` and `topic_seqend` fields.

In the next following vresion of the trading system, the maximum range will be set.

If a request can be processed, the client will receive `TopicReport` and after that should expect data messages. After data transfer is completed, the client will also receive `TopicReport`.

If a request is incorrect or cannot be processed, the gateway will respond with `TopicReject`.

If you want to request a new topic, wait until you have received all messages, related to the previous topic request, to avoid network overload.

Table 50. Format of message `TopicRequest`: msgid=301, size=101

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	topic	ascii64	Topic ID
84	topic_seq	int8	First number of requested messages

Offset	Field	Datatype	Description
92	topic_seqend	int8	Last number of requested messages
100	mode	int1	Broadcast mode. Value: `0` (DATA_SLICE): snapshot

4.1.9. Report on rejecting request

If the client's request is incorrect or cannot be processed, the gateway will send the `TopicReject` message. The reason for rejection is specified in the `reason` field.

The message includes reference fields `topic_lastseq` (the number of the last message generated in the topic) and `topic_lastseqsent` (the number of the last message sent to the client).

Table 51. Format of message `TopicReject`: msgid=402, size=142

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	topic	ascii64	Topic ID
110	topic_id	int4	Numerical ID of topic
114	status	int2	Status of data transfer. Value: `0` (DATA_SLICE): snapshot transfer
116	reason	int2	Reason for rejection. Values: • `1` (BAD_TOPIC): invalid topic identifier; • `4` (DATA_NOT_AVAILABLE): data not available; • `5` (DUPLICATE_REQUEST): repeated request; • `6` (BAD_SEQ): non-existent number in topic; • `7` (BAD_MODE): invalid mode
118	topic_firstseq	int8	Number of first available message
126	topic_lastseq	int8	Number of the last message generated in the topic
134	topic_lastseqsent	int8	Number of the last message sent to the client

4.1.10. Report on executing request

The client will receive notification `TopicReport` in the following cases:

- successful execution of the data request;
- completion of snapshot transmission.

The message includes reference fields `topic_lastseq` (the number of the last message generated in the topic) and `topic_lastseqsent` (the number of the last message sent to the client).

Table 52. Format of message `TopicReport`: msgid=401, size=134

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header

Offset	Field	Datatype	Description
46	topic	ascii64	Topic ID
110	topic_id	int4	Numerical ID of topic
114	status	int2	Status of data transfer. Value: 0 (DATA_SLICE): snapshot transfer
116	marker	int2	Indicator of start and finish of data transfer. Values: • 0 (START): start of data transfer; • 2 (SLICE_END): snapshot transfer completed
118	topic_lastseq	int8	Number of the last message generated in the topic
126	topic_lastseqsent	int8	Number of the last message sent to the client

Appendix A. Error codes

Table 53. Error codes list

Code	Description
0	Ok
5	Missed tag.
100	Filled excess tag.
999	Internal error.
1000	Incorrect login.
1001	Incorrect instrument.
1002	Incorrect client ID.
1003	Invalid member_id.
1004	Invalid account.
1005	Incorrect client group.
1006	Incorrect exchange.
1007	Instrument not traded.
1008	Invalid routing options.
1100	Invalid order direction.
1101	Incorrect price.
1102	Incorrect price_extra.
1103	Incorrect amount.
1104	Incorrect amount_extra.
1105	Invalid order type.
1106	Invalid time_in_force.
1107	Invalid passive_only.
1108	Invalid auto_cancel.
1109	Invalid flags.
1110	Invalid mode.
1111	Incorrect clorder_id.
1112	Incorrect orig_clorder_id.
1113	Invalid prime_exchange.
1114	Invalid date_expire.
1115	Invalid comment.
1116	Invalid level.

Code	Description
1117	Invalid trade_mode.
1200	Invalid segment.
1201	Incorrect extra1.
1202	Incorrect OTC code for negotiated trade initiator.
1203	Incorrect OTC code for counter party.
1204	Invalid order_type for this instrument.
1205	Order_type not supported by exchange.
1206	Invalid order_type for Client ID.
1207	Incorrect price for this order_type.
1208	Incorrect amount_extra for this order_type.
1209	Invalid time_in_force for this order_type.
1210	Invalid flags for this order_type.
1211	Invalid instrument for replacement mode.
1212	Invalid member_id for replacement mode.
1213	Invalid client_id for replacement mode.
1214	Invalid account for replacement mode.
1215	Invalid parameters of rejected counter order.
1216	Invalid replacement parameters.
1217	Invalid time_in_force for this instrument.
1218	Invalid replacement mode for this login.
1219	Invalid flags for this instrument.
1300	Both orig_clorder_id and order_id filled.
1301	Duplicate clorder_id.
1302	Price exceeds limits.
1303	Order type not supported for this client ID.
1304	Order type not supported by exchange.
1305	Invalid prime_exchange for this instrument.
1306	Liquidity pool unavailable for client ID.
1307	Invalid order_type for this instrument.
1308	User has no permissions to cancel orders of account specified.
1309	User has no permissions to replace orders of account specified.
1310	User has no permissions to reject this order.

Code	Description
1311	Order currently being replaced.
1312	Order sent before system crash, but received after recovery.
1313	Limitation not available for this instrument.
1314	User has no permissions to use this mode.
1315	This exchange is prohibited for clearing member.
1316	This exchange is prohibited for trade member.
1317	Order submission via the login is blocked.
1318	Order submission via the login is blocked for the client code.
1319	Order submission via the login is blocked for the TCA.
1400	Instrument not available for market maker.
1401	No permissions to trade this instrument.
1402	No permissions to indicate 'No matching another market maker's orders'.
1403	Client has no permissions to trade with using this account.
1404	Liquidity pool not available for this smart order router.
1500	Trade engine IDs (te_id) do not match.
1501	Incorrect te_id.
1502	Request received during the limited margin update.
1700	User has no permission for limited margin service.
1701	Client has no permissions for limited margin service.
1702	Client group has no permissions for limited margin service.
1703	Account has no permissions for limited margin service.
1704	Main account has no permissions for limited margin service.
1710	Invalid parameters for limited margin of client.
1711	Invalid parameters for limited margin of client group.
1712	Invalid parameters for limited margin of account.
1713	Invalid parameters for limited margin of main account.
1714	Request for limited margin update for client received when the previous request still processing.
1715	Request for limited margin update for client group received when the previous request still processing.
1716	Request for limited margin update for TCA received when the previous request still processing.
1717	Request for limited margin update for principal TCA received when the previous request still processing.
1720	Incorrect limit for limited margin.
1721	Incorrect instrument limit for limited margin.

Code	Description
1722	Incorrect order limit for limited margin.
1723	Incorrect extra limit for limited margin.
1750	Insufficient limit for limited margin of client.
1751	Insufficient instrument limit for limited margin of client.
1752	Insufficient order limit for limited margin of client.
1753	Insufficient extra limit for limited margin of client.
1754	Insufficient limit for limited margin of client group.
1755	Insufficient instrument limit for limited margin of client group.
1756	Insufficient order limit for limited margin of client group.
1757	Insufficient extra limit for limited margin of client group.
1758	Insufficient limit for limited margin of account.
1759	Insufficient instrument limit for limited margin of account.
1760	Insufficient order limit for limited margin of account.
1761	Insufficient extra limit for limited margin of account.
1762	Insufficient limit for limited margin of main account.
1763	Insufficient instrument limit for limited margin of main account.
1764	Insufficient order limit for limited margin of main account.
1765	Insufficient extra limit for limited margin of main account.
1766	The client has active orders of limited margin.
1767	The client group has active orders of limited margin.
1768	The TCA has active orders of limited margin.
1769	The principal TCA has active orders of limited margin.
1770	Limited margin suspended for client.
1771	Limited margin suspended for client group.
1772	Limited margin suspended for account.
1773	Limited margin suspended for main clearing account.
1780	Invalid liquidity pool for limited margin service.
1800	Incorrect yield type specified.
1801	Incorrect yield conversion direction specified.
1980	Invalid stages in info field.
2100	Account does not belong to member_id.
2200	No permissions to submit trading instructions.

Code	Description
2201	Client group level prohibition is set.
2202	Trade member level prohibion is set.
2203	Clearing member prohibion is set.
2204	Trade administrator level prohibion is set.
2300	No permissions to place an unsecured order.
2400	No permissions to cancel order.
2600	No permissions to set limit for clearing account.
2601	No permissions to set limits for client ID.
2602	No permissions to set limits for client group.
2603	Invalid type.
2604	Invalid value.
2605	Ambiguous type.
2700	Client ID has insufficient funds.
2701	Client ID has insufficient assets.
2702	Client group has insufficient funds.
2703	Client group has insufficient assets.
2704	Account has insufficient funds.
2705	Account has insufficient assets.
2706	Main clearing account has insufficient funds.
2707	Main clearing account has insufficient assets.
2708	Clearing member has insufficient funds.
2709	Insufficient blocked assets.
3000	Market or IOC order expired after no trades.
3001	Order canceled after no trades, to avoid a cross trade.
3002	Order canceled after no trades, to avoid a crossed book.
3003	Client order not found.
3004	Instrument trading suspended.
3100	TCA of maker and that of taker have no conversion bank indicator.
3911	Incorrect te_id.
4000	ECN not available or no liquidity pool available.
4001	The specified liquidity pool not available.
4002	Order forcedly routed to a liquidity pool after rejected by risk management at the trading system.

Code	Description
4003	Client ID not registered at all the available liquidity pools.
4004	Client ID not registered at the trading system.
4005	Client ID not registered at liquidity pool.
4006	Order cannot be routed to any liquidity pool.
4100	Order pending cancel.
4200	Invalid client for TCA registered at liquidity pool.
4201	Invalid TCA for liquidity pool.
5000	Invalid application message type.
5001	Invalid routing_dest.
5002	Invalid message type for this login.
5003	Login has no permissions to submit such instruction.
5200	User already logged in.
5201	Discovery service settings timeout.
5202	Incorrect heatbeat_ms.
5203	Incorrect user ID / password.
5204	Incorrect message sequence number.
5205	Invalid session message type.
5206	User not logged in.
5207	Another resend request processing in progress.
5208	Incorrect range limit.
5209	Invalid reset_seq.
5210	Requested messages range excess.
5211	Invalid session message size.
5212	Disconnected by the operator.
5300	Invalid topic.
5301	Snapshot with updates has already been requested.
5302	Snapshot with updates has not been requested.
5303	Requested data not available.
5304	Another request processing in progress.
5400	Reset_seq indicated, but seqnums cannot be reset.
5401	Number of messages exceeded limit.
5601	Both account and parties filled.

Code	Description
7000	Order canceled before sending to ASTS.
7001	Order canceled as no answer received.

Also you can get errors come in range —11000-11999. These are the error codes returned by the trading system of the Moscow stock exchange (ASTS). To get the ASTS error id , you need to subtract 11000 from the internal error id. The description of these errors, a client can get from the ASTS documentation.

Appendix B. Revision history

Version 1.13.0 24 December 2015

1. The is_test field is added in Currency, Issue, Spot, Futures, and Bond messages.
2. In Instrument message, the is_test, te_id, and be_mode fields are added, the reserved removed and the msgid changed.
3. In the Underlying component, the flags field is added and the size of the qty field is altered.

Version 1.12.0 10 November 2015

The broadcast of the CurrentPriceOfMarket topic started.

Version 1.11.1 14 October 2015

The datatype for the type=76 value is altered in the Commons update.

Version 1.11.0 1 October 2015

The size of the Underlying component is altered, because the `qty` datatype is changed.

Version 1.10.0 2 July 2015

1. New value 76 of the `type` field in the Commons Update added.
2. The format of Instrument message amended—size of `trade_mode_id` reduced to 2 bytes and `reserved` field added.